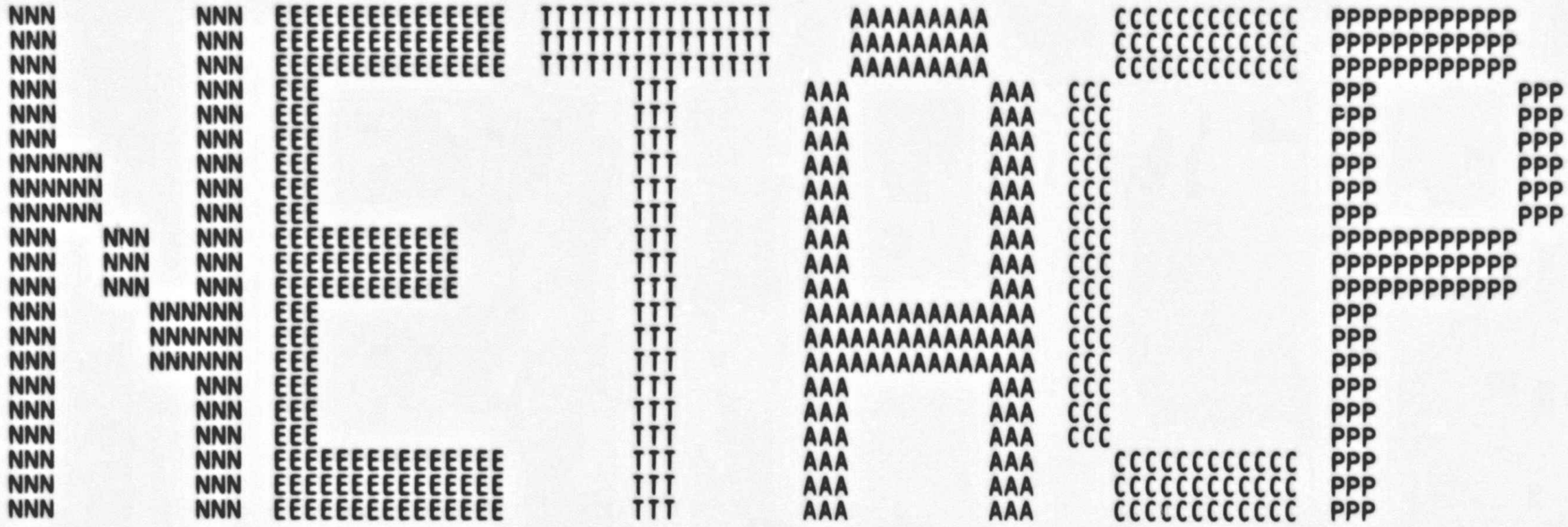
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPP
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPP
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNNNNN NNN EEE TTT AAA AAA CCC PPP PPP
NNNNNN NNN EEE TTT AAA AAA CCC PPP PPP
NNNNNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN NNN NNN EEEEEEEEEEE TTT AAA AAA CCC PPPPPPPPPPP
NNN NNN NNN EEEEEEEEEEE TTT AAA AAA CCC PPPPPPPPPPP
NNN NNN NNN EEEEEEEEEEE TTT AAA AAA CCC PPPPPPPPPPP
NNN NNNNNN EEE TTT AAAAAAAAAAAAAAA CCC PPP
NNN NNNNNN EEE TTT AAAAAAAAAAAAAAA CCC PPP
NNN NNNNNN EEE TTT AAAAAAAAAAAAAAA CCC PPP
NNN NNN EEE TTT AAA AAA CCC PPP
NNN NNN EEE TTT AAA AAA CCC PPP
NNN NNN EEE TTT AAA AAA CCC PPP
NNN NNN EEEEEEEEEEEEEEE TTT AAA AAA CCCCCCCCCCC PPP
NNN NNN EEEEEEEEEEEEEEE TTT AAA AAA CCCCCCCCCCC PPP
NNN NNN EEEEEEEEEEEEEEE TTT AAA AAA CCCCCCCCCCC PPP

FILEID**NETDRVNSP

```
NN      NN  EEEEEEEEEE  TTTTTTTTTT  DDDDDDDD    RRRRRRRR    VV      VV  NN      NN    SSSSSSSS  PPPPPPPP
NN      NN  EEEEEEEEEE  TTTTTTTTTT  DDDDDDDD    RRRRRRRR    VV      VV  NN      NN    SSSSSSSS  PPPPPPPP
NN      NN  EE              TT      DD      DD  RR      RR  VV      VV  NN      NN  SS          PP      PP
NN      NN  EE              TT      DD      DD  RR      RR  VV      VV  NN      NN  SS          PP      PP
NNNN    NN  EE              TT      DD      DD  RR      RR  VV      VV  NNNN    NN  SS          PP      PP
NNNN    NN  EE              TT      DD      DD  RR      RR  VV      VV  NNNN    NN  SS          PP      PP
NN  NN  NN  EEEEEEEE        TT      DD      DD  RRRRRRRR    VV      VV  NN  NN  NN    SSSSSS    PPPPPPPP
NN  NN  NN  EEEEEEEE        TT      DD      DD  RRRRRRRR    VV      VV  NN  NN  NN    SSSSSS    PPPPPPPP
NN    NNNN  EE              TT      DD      DD  RR  RR      VV      VV  NN    NNNN          SS  PP
NN    NNNN  EE              TT      DD      DD  RR  RR      VV      VV  NN    NNNN          SS  PP
NN      NN  EE              TT      DD      DD  RR    RR      VV  VV    NN      NN          SS  PP
NN      NN  EE              TT      DD      DD  RR    RR      VV  VV    NN      NN          SS  PP
NN      NN  EEEEEEEEEE      TT      DDDDDDDD    RR      RR      VV      NN      NN  SSSSSSSS  PP
NN      NN  EEEEEEEEEE      TT      DDDDDDDD    RR      RR      VV      NN      NN  SSSSSSSS  PP

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

Table of contents

(2)	41	MODIFICATION HISTORY	
(3)	79	DECLARATIONS	
(5)	166	NSP MESSAGE FORMAT	
(6)	217	FLOW CONTROL OVERVIEW	
(7)	252	LSB State Variable Description	
(9)	405	NET$SETUP_RUN	- Setup XWB for the RUN state
(10)	577	NET$ALTENTRY	- Driver alternate entry point
(11)	613	NET$FDT_RCV	- Process IO$_READxBLK requests
(11)	614	NET$FDT_XMT	- Process IO$_WRITExBLK requests
(20)	841	NET$UNSOL_INTR	- Receive from Transport Layer
(21)	1036	ACT$RTS_NLT	- Return to sender as "no-link-terminate"
(22)	1216	ACT$RCV_CC	- Respond to a received Connect Confirm message
(22)	1217	ACT$RCV_CA	- Respond to Connect Acknowledge
(22)	1218	ACT$RCV_CI	- Process received Connect Initiate message
(23)	1312	PRS_CHR	- Get characteristics from Connect message
(27)	1567	ACT$RCV_RTS	- Receive CI message being "returned to sender"
(27)	1568	ACT$RCV_Dx	- Recieve DI/DC message
(27)	1569	ACT$ABORT	- Disconnect or abort a link
(27)	1570	ACT$CANLNK	- Disconnect link due to user's $CANCEL
(28)	1648	ACT$RCV_DTACK	- DATA ACK message processing
(28)	1649	ACT$RCV_LIACK	- INT/LI ACK message processing
(28)	1650	NET$PIG_ACK	- Common piggy-backed ACK processing
(29)	1760	PROC_DTACK	- Process of DATA ACK
(30)	1813	PROC_LIACK	- Process INT/LS ACK
(31)	1870	NET$ACK_XMT_SEGS	- ACK Xmt Segs, Complete User Xmt IRP's
(34)	1981	ACT$RCV_LI	- Receive INT/LS message
(35)	2143	CHK_INT_AVL	- Conditionally set XWB$V_FLG_IAVL
(35)	2144	CHK_INT_AVL_R8	- Conditionally set XWB$V_FLG_IAVL
(36)	2175	SHRINK_XPW	- Shrink the DATA transmit-packet-window
(36)	2176	NEW_DATA_FLOW	- React to flow control msg
(37)	2255	CALC_HXS...	- Calc 'highest xmt seg sendable'
(40)	2422	ACT$RCV_DATA	- Process rcv'd DATA message
(56)	3163	CLONE_RCV_CXB	- Clone a copy of a rcv'd CXB
(59)	3246	NSP$SOLICIT	- Solicit permission to transmit
(60)	3407	BLD_DISPATCH	- Dispatch to build message
(61)	3496	BLD_CD	- Build Connect/Disconnect messages
(61)	3497	BLD_CI	- Build a CI msg from XWB contents
(61)	3498	BLD_CA	- Build a CA msg from XWB contents
(61)	3499	BLD_CC	- Build a CC msg from XWB contents
(61)	3500	BLD_DI	- Build a DI msg from XWB contents
(61)	3501	BLD_DC	- Build A DC msg from XWB contents
(62)	3654	BLD_LIACK	- Build a INT/LS ACK message
(62)	3655	BLD_DTACK	- Build a DATA ACK message
(62)	3656	BLD_LI	- Build INT/LS message
(62)	3657	BLD_DAT	- Build DATA message
(63)	3839	GET_XMT_CXB	- Get xmt CXB while in FDT context
(64)	3896	GET_XMT_BUF	- Get xmt buffer while in fork context
(65)	3945	NET$IO_STATUS	- Receive xmit status from Transport layer
(65)	3946	NET$CCS_IOSTAT	- Receive xmit status for Phase II CC message
(66)	3993	NET$TIMER	- Process NETDRIVER clock tick
(67)	4214	TIMED_SEG_ACKED	- Timed segment has been ACK'd

```
0000     1
0000     2          .TITLE  NETDRVNSP - DECnet NSP module for NETDRIVER
0000     3          .IDENT  'V04-000'
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27
0000    28 ;++
0000    29 ; FACILITY:     DECnet, Executive
0000    30 ;
0000    31 ; ABSTRACT:
0000    32 ;               This module implements that NSP layer of NETDRIVER.  NSP
0000    33 ;               is the protocol spoken over logical-links.  The NSP layer
0000    34 ;               is sandwiched between the Session and Routing layers; each
0000    35 ;               of which is implemented in a separate NETDRIVER module.
0000    36 ;
0000    37 ; ENVIRONMENT:  Standard driver environment
0000    38 ;--
0000    39
```

```
0000     41          .SBTTL  MODIFICATION HISTORY
0000     42 ;
0000     43 ; AUTHOR:       Alan D. Eldridge,  CREATION DATE: 11-Mar-1982
0000     44 ;
0000     45 ; MODIFIED BY:
0000     46 ;
0000     47 ;       V03-033 ADE0043         A. Eldridge             10-Aug-1984
0000     48 ;               Don't update remote node address in XWB since that address is
0000     49 ;               used as part of the NETACP hashing to locate the node counter
0000     50 ;               block, etc.
0000     51 ;
0000     52 ;       V03-032 ADE0042         A. Eldridge             21-Jul-1984
0000     53 ;               Fix race condition in receiver which was causing segments to
0000     54 ;               be copied out of order.
0000     55 ;
0000     56 ;       V03-031 ADE0041         A. Eldridge             28-Jun-1984
0000     57 ;               Move window and buffer control parameters to storage area so
0000     58 ;               that they can be more easily played with via PATCH for
0000     59 ;               experimentation.
0000     60 ;
0000     61 ;               Don't allow ACK delay on data message exactly halfway into
0000     62 ;               the pipeline.  This allows overlap of the data and returning
0000     63 ;               ACK message streams.
0000     64 ;
0000     65 ;               Change max pipeline window to 40 (was 7).  Modify window
0000     66 ;               adjustment algorithms.
0000     67 ;
0000     68 ;               Fix bug in INTerrupt message FDT routine that called
0000     69 ;               CHK_INT_AVL with the wrong LSB pointer in R2.
0000     70 ;
0000     71 ;       V03-030 ADE0040         A. Eldridge             10-Sep-1983
0000     72 ;               Major rewrite to build data segments as needed (rather than
0000     73 ;               just at FDT time) by using kernel mode AST's to nibble away at
0000     74 ;               the user buffer.
0000     75 ;
0000     76 ;
0000     77
```

```
0000     79          .SBTTL  DECLARATIONS
0000     80 ;
0000     81 ; INCLUDE FILES:
0000     82 ;
0000     83          $AQBDEF
0000     84          $ACBDEF
0000     85          $CCBDEF
0000     86          $CRBDEF
0000     87          $CXBDEF
0000     88          $DDBDEF
0000     89          $DPTDEF
0000     90          $DRDEF
0000     91          $DYNDEF
0000     92          $IPLDEF
0000     93          $IRPDEF
0000     94          $IODEF
0000     95          $JIBDEF
0000     96          $MSGDEF
0000     97          $PCBDEF
0000     98          $PHDDEF
0000     99          $PRDEF
0000    100          $RSNDEF
0000    101          $SSDEF
0000    102          $TQEDEF
0000    103          $UCBDEF
0000    104          $VADEF
0000    105          $VECDEF
0000    106
0000    107          $ICBDEF
0000    108          $IDBDEF
0000    109          $LLIDEF
0000    110          $LTBDEF
0000    111          $RCBDEF
0000    112
0000    113          $NETSYMDEF
0000    114          $NETUPDDEF
0000    115          $NSPMSGDEF
0000    116
0000    117          $CXBEXTDEF              ; NETDRIVER CXB extensions
0000    118          $XWBDEF                 ; XWB and LSB definitions
0000    119
0000    120
```

```
                     0000   122 ;
                     0000   123 ; EQUATED SYMBOLS
                     0000   124 ;
00000000             0000   125 P1       = 0                     ; QIO P1 parameter offset from AP
00000004             0000   126 P2       = 4                     ; QIO P2 parameter offset from AP
                     0000   127
00000084             0000   128 NDC      = XWB$Z_NDC             ; Shortened symbol name for counter offset
                     0000   129
00000048             0000   130 IRP$L_SES_BUF   == IRP$L_SEGVBN
00000004             0000   131 IRP$B_QUO       == 4
00000005             0000   132 IRP$B_CXBCNT    == 5
                     0000   133
                     0000   134 .iif ndf,IRP$Q_STATION, IRP$Q_STATION = 8+IRP$L_IOST1
                     0000   135 .iif ndf,IO$V_MULTIPLE, IO$V_MULTIPLE = 1+IO$V_INTERRUPT
                     0000   136 .iif ndf,IO$M_MULTIPLE, IO$M_MULTIPLE = 1@IO$V_MULTIPLE
                     0000   137
00000020             0000   138 NSP$C_ADJ_XPW   = 32
00000028             0000   139 NSP$C_MAX_XPW   = 40
00000007             0000   140 NSP$C_MAX_RBF   =  7
00000005             0000   141 NSP$C_R_CXBTHR  =  5
                     0000   142
0000000D             0000   143 NSP$V_ACK_XCH   = 13
0000000E             0000   144 NSP$V_SEQ_NAR   = 14
00004000             0000   145 NSP$M_SEQ_NAR   = 1@NSP$V_SEQ_NAR
00000007             0000   146 NSP$V_DATA_NAR  = NSP$V_DATA_EOM + 1
00000080             0000   147 NSP$M_DATA_NAR  = NSP$M_DATA_EOM * 2
                     0000   148
00000002             0000   149 NSP$C_INF_V40   = NSP$C_INF_V33
00000068             0000   150 NSP$C_MSG_CR    = ^X<68>                ; Retransmitted Connect Initiate
                     0000   151                                         ;; put in library
                     0000   152 ;
                     0000   153 ; MACROS:
                     0000   154 ;
                     0000   155
                     0000   156 ;
                     0000   157 ;  Bit definition macro
                     0000   158 ;
                     0000   159 .MACRO  BITDEF  BLK,SYM,BITVAL
                     0000   160
                     0000   161         'BLK'$V_'SYM' = BITVAL
                     0000   162         'BLK'$M_'SYM' = 1@<BITVAL>
                     0000   163 .ENDM
                     0000   164
```

```
0000   166          .SBTTL  NSP MESSAGE FORMAT
0000   167
0000   168 ;++
0000   169 ;
0000   170 ;  <0eb0 0000><4b_LINK>{2b_ACK}<2b_SEG><DATA>                          DATA MSG
0000   171 ;  <0011 0000><4b_LINK>{2b_ACK}<2b_SEG><u16_DATA>                     INT. MSG
0000   172 ;  <0001 0000><4b_LINK>{2b_ACK}<2b_SEG><2b_FLOW>                     L.S. MSG
0000   173 ;
0000   174 ;  <0000 0100><4b_LINK><2b_ACK>                                     DATA ACK
0000   175 ;  <0001 0100><4b_LINK><2b_ACK>                                     OTH. ACK
0000   176 ;  <0010 0100><2b_DST>                                              CA
0000   177 ;
0000   178 ;  <0001 1000><2k_0><2b_SRC><1b_SRV><1b_INFO><2b_SEGSIZ><CTL>      CI
0000   179 ;  <0101 1000><2k_0><2b_SRC><1b_SRV><1b_INFO><2b_SEGSIZ><CTL>      CR
0000   180 ;  <0010 1000><4b_LINK><1b_SRV><1b_INFO><2b_SEGSIZ><i16_DATA>      CC
0000   181 ;  <0011 1000><4b_LINK><2b_REA><i16_DATA>                           DI
0000   182 ;  <0100 1000><4b_LINK><2b_REA>                                     DC
0000   183 ;  <0100 1000><2b_DST><2k_0><2k_1>                                  CT
0000   184 ;  <0100 1000><4b_LINK><2k_42>                                      DT
0000   185 ;  <0100 1000><4b_LINK><2k_41>                                      NLT
0000   186 ;
0000   187 ;  <0101 1000>------                                               START
0000   188 ;
0000   189 ;  <4b_LINK   ::=  <2b_DST><2b_SRC>              link address, not = 0
0000   190 ;  <2b_ACK    ::=  <1001><12 bit seg number>     if NAK
0000   191 ;                  <1000><12 bit seg number>     if ACK
0000   192 ;  <2b_SEG>   ::=  <0000><12 bit seg number>
0000   193 ;  <2b_FLOW>  ::=  <00000><1 bit subchannel><2 bit mode><1 byte count>
0000   194 ;                        0 => data          00 => no change
0000   195 ;                        1 => interrupt     01 => stop
0000   196 ;                                           10 => start
0000   197 ;
0000   198 ;  <1b_SRV>   ::=  <00000001>                  if no flow control
0000   199 ;                  <00000101>                  if segment flow control
0000   200 ;                  <00001001>                  if message flow control
0000   201 ;  <1b_INFO>  ::=  <00000001>                  if NSP V3.1
0000   202 ;                  <00000000>                  if NSP V3.2
0000   203 ;
0000   204 ;
0000   205 ;  <CTL>       ::=  <DNAME><SNAME><000000da><ACCOUNT><i16_DATA>
0000   206 ;                                              if a      if d
0000   207 ;                       <DNAME> ::= <NAME>
0000   208 ;                       <SNAME> ::= <NAME>
0000   209 ;                       <NAME>  ::= <1k_0><1b_objtyp>  objtyp not= 0
0000   210 ;                                   <1k_1><1k_0><i16_desc>
0000   211 ;                                   <1k_2><1k_0><2b_gcod><2b_ucod><i12_desc>
0000   212 ;                       <ACCT>  ::= <i39_id><i39_psw><i39_acc>
0000   213 ;
0000   214 ;
0000   215 ;--
```

```
0000   217          .SBTTL  FLOW CONTROL OVERVIEW
0000   218
0000   219 ;
0000   220 ;  The DATA subchannel transmitter is either message, segment, or null
0000   221 ;  flowed controlled by the remote receiver.  For details consult the
0000   222 ;  NSP Functional Spec.  Briefly, the rules are as follows:
0000   223 ;
0000   224 ;  - Null Flow Control
0000   225 ;
0000   226 ;    There is no flow control.  Backpressure is the only way that the
0000   227 ;    receiver can force the transmitter to stop transmitting.
0000   228 ;
0000   229 ;  - Message Flow Control
0000   230 ;
0000   231 ;    The receive increments the flow control variable once for each
0000   232 ;    message which it may receive.   It may never decrement it.  This
0000   233 ;    value must never exceed 127.
0000   234 ;
0000   235 ;  - Segment Flow Control
0000   236 ;
0000   237 ;    The adds the flow control value to the current flow control
0000   238 ;    variable, this may increment it (not past 127) or decrement it (not
0000   239 ;    below zero).
0000   240 ;
0000   241 ;
0000   242 ;  Managing DATA Transmission Control Variables
0000   243 ;
0000   244 ;  To control the xmitter, the following parameters are defined.  Each
0000   245 ;  is a signed 12 bit number referring to an NSP message sequence number.
0000   246 ;  Since NSP may not "pipeline" more than 2098 messages on any given
0000   247 ;  subchannel, sequence number A is less than sequence number B if
0000   248 ;  B-A < 2098 (mod 4096).
0000   249 ;
0000   250 ;
```

```
0000    252          .SBTTL  LSB State Variable Description
0000    253
0000    254 ;
0000    255 ;
0000    256 ;  LSB Transmitter segment number variables:
0000    257 ;
0000    258 ;       HAR     Highest ACK Received.  This value is increased upon receiving
0000    259 ;               an ACK for a transmitted segment.  It is never decreased.
0000    260 ;
0000    261 ;       HXS     Highest Xmt-able Segment.  This value represents the highest
0000    262 ;               segment number which is both currently queued and allowed to
0000    263 ;               be sent according to the remote receivers flow control credits,
0000    264 ;               flow control type (message, segment, none), and the transmit-
0000    265 ;               packet window.  It is independent of the current backpressure
0000    266 ;               setting.  It may be increased whenever:
0000    267 ;
0000    268 ;                 - a new message segment is Queued from by the session layer.
0000    269 ;                 - positive flow control credits are received.
0000    270 ;                 - the transmit-packet-window is opened.
0000    271 ;
0000    272 ;               It may be decreased whenever:
0000    273 ;
0000    274 ;                 - negative flow control credits are received.
0000    275 ;                 - the transmit-packet-window is opened.
0000    276 ;                 - the session layer does a $CANCEL (not yet supported since
0000    277 ;                   this currently breaks the logical-link).
0000    278 ;
0000    279 ;
0000    280 ;       LNX     Last Number Xmt'd.  This is the number of the last segment
0000    281 ;               sent to the Routing layer for transmission.  It is decreased
0000    282 ;               whenever messages need to be retransmitted due to a timeout
0000    283 ;               or a received NAK.  It is increased whenever a segment is
0000    284 ;               sent to the Routing layer for transmission.
0000    285 ;
0000    286 ;       LUX     Last Used Xmt-number.  This is the number of the last segment
0000    287 ;               number assigned to a segment.  On the DATA channel, segment
0000    288 ;               numbers are assigned to the buffered data segements as they are
0000    289 ;               built at FDT (or ALTSTART) time.  On the LS/INTERRUPT channel
0000    290 ;               they are assigned as a message is sent for the first time.
0000    291 ;               The latter technique could be used on the DATA channel as well,
0000    292 ;               but it would make the calculation of HXS (and this happens
0000    293 ;               very frequently) inefficient.
0000    294 ;
0000    295 ;               Hence, this variable increases whenever a new segement number
0000    296 ;               is assigned and never decreases.
0000    297 ;
0000    298 ;
0000    299 ;       HAA     Highest ACK Acceptable.  It is increased when a message is
0000    300 ;               transmitted and the new LNX is greater than the old HAA.  It is
0000    301 ;               decreased if the trasnmitter is "segment" flow controlled and
0000    302 ;               receives negative flow credits which caused the old HAA to be
0000    303 ;               no longer flow controlled.
0000    304 ;
0000    305 ;
0000    306 ;
0000    307 ;  LSB Rules
0000    308 ;
```

```
0000    309 ;       HAR             0a.  Increases but never decreases
0000    310 ;       HXS             0b.  Increases and decreases
0000    311 ;       LNX             0c.  Increases and decreases
0000    312 ;       LUX             0d.  Increases but never decreases
0000    313 ;       HAA             0e.  Increases and decreases
0000    314 ;
0000    315 ;       HAR leq HXS     1a.  Never advance HAR beyond HXS without advancing HXS
0000    316 ;                       1b.  Never shrink  HXS below  HAR
0000    317 ;
0000    318 ;       HAR leq LNX     2a.  Never advance HAR beyond LNX without advancing LNX
0000    319 ;                       2b.  Never shrink  LNX below  HAR
0000    320 ;
0000    321 ;       HAR leq LUX     3a.  Never advance HAR beyond LUX (done via 1a. + 5a.)
0000    322 ;                       3b.  Never shrink  LUX below  HAR (done via 0d.)
0000    323 ;
0000    324 ;       HXS geq LNX     4a.  Never shrink  HXS below  LNX without shrinking LNX
0000    325 ;                       4b.  Never advance LNX beyond HXS
0000    326 ;
0000    327 ;       HXS leq LUX     5a.  Never advance HXS beyond LUX
0000    328 ;                       5b.  Never shrink  LUX below  HXS (done via 0d.)
0000    329 ;
0000    330 ;       LNX leq LUX     6a.  Never advance LNX beyond LUX (done via 4b. + 5a.)
0000    331 ;                       6b.  Never shrink  LUX below  LNX (done via 0d.)
0000    332 ;
0000    333 ;       HAA geq HAR     7a.  Never advance HAR beyond HAA
0000    334 ;                       7b.  Never shrink  HAA below  HAR (done via 1b.)
0000    335 ;
0000    336 ;       HAA --- HXS     8a.  Can be less than, equal to, or greater than
0000    337 ;
0000    338 ;       HAA geq LNX     9a.  Never advance LNX beyond HAA without advancing HAA
0000    339 ;                       9b.  Never shrink  HAA below  LNX without shrinking LNX
0000    340 ;                                                         (done via 4a.)
0000    341 ;
0000    342 ;       HAA leq LUX     10a. Never advance HAA beyond LUX (done via 6a. + 9a.)
0000    343 ;                       10b. Never shrink  LUX below  HAA (done via 0d.)
0000    344 ;
0000    345 ;
0000    346 ;  LSB Receiver segment number variables:
0000    347 ;
0000    348 ;       HAX leq HNR     11a. Never advance HAX beyond HNR
0000    349 ;                            (HAX never shinks)
0000    350 ;
0000    351 ;
0000    352
```

```
                                                K 2
00000000  0000  354          .PSECT  $$$115_DRIVER,LONG,EXE,RD,WRT
          0000  355
      20  0000  356 NSP$B_ADJ_XPW:: .BYTE   NSP$C_ADJ_XPW
      28  0001  357 NSP$B_MAX_XPW:: .BYTE   NSP$C_MAX_XPW
      07  0002  358 NSP$B_MAX_RBF:: .BYTE   NSP$C_MAX_RBF
      05  0003  359 NSP$B_R_CXBTHR::.BYTE   NSP$C_R_CXBTHR
          0004  360
          0004  361 ;
          0004  362 ;
          0004  363 ;  The following table is used to map a received message into an event.  It
          0004  364 ;  is ordered according to the most likely received event and is terminated
          0004  365 ;  with a longword of zero.
          0004  366 ;
          0004  367 ;  It also contains miscellaneous information -- what the minimum size of the
          0004  368 ;  message is, and whether or not the message size is fixed or variable
          0004  369 ;
          0004  370 ;
00000000  0004  371 RCVMAP_B_MSG = 0
00000001  0004  372 RCVMAP_B_SIZ = 1
00000002  0004  373 RCVMAP_B_EVT = 2
0000FFFF  0004  374 RCVMAP_C_END = ^X<FFFF>                     ; MSG code used to terminate table
          0004  375
          0004  376 .MACRO  MAP_RCV_MSG     MSG,FIXED,MIN_SIZ
          0004  377
          0004  378          .BYTE   NSP$C_MSG_'msg'           ; Message type
          0004  379          .IF  NB,FIXED                     ; 1 if fixed sized message, else 0
          0004  380               .BYTE   min_siz              ; fixed - enter minimum msg size
          0004  381          .IFF                              ;
          0004  382               .BYTE   -min_siz             ; variable - enter negative min msg siz
          0004  383          .ENDC                             ;
          0004  384          .BYTE   NETEVT$_'msg'             ; Event code
          0004  385
          0004  386 .ENDM   MAP_RCV_MSG
          0004  387
          0004  388 NET$AT_RCVMSG:
          0004  389
          0004  390 MAP_RCV_MSG     DATA,     ,  7            ; Data message
          0007  391 MAP_RCV_MSG     DTACK,    ,  7            ; Data Ack
          000A  392 MAP_RCV_MSG     LS,       ,  9            ; Link service message
          000D  393 MAP_RCV_MSG     LIACK,    ,  7            ; Link service/Interrupt Ack
          0010  394 MAP_RCV_MSG     INT,      ,  7            ; Interrupt message
          0013  395 MAP_RCV_MSG     CI,       , 10            ; Connect Initiate
          0016  396 MAP_RCV_MSG     CA,     F,  3            ; Connect Ack
          0019  397 MAP_RCV_MSG     CC,       ,  9            ; Connect Confirm
          001C  398 MAP_RCV_MSG     DI,       ,  7            ; Disconnect Initiate
          001F  399 MAP_RCV_MSG     DC,     F,  7            ; Disconnect Confirm
          0022  400
FFFFFFFF  0022  401                    .LONG   -1             ; Terminate the table
          0026  402 .ALIGN  LONG
          0028  403
```

```
                     0028   405          .SBTTL  NET$SETUP_RUN   - Setup XWB for the RUN state
                     0028   406 ;+
                     0028   407 ;
                     0028   408 ;  INPUTS:  R5  XWB address
                     0028   409 ;           R0  Scratch
                     0028   410 ;
                     0028   411 ;  OUTPUTS: R0  Low bit set
                     0028   412 ;
                     0028   413 ;           All other registers are preserved.
                     0028   414 ;
                     0028   415 ;
                     0028   416 ;-
                     0028   417 NET$SETUP_RUN::                                  ; Setup XWB for RUN state
                     0028   418
          07DE 8F    BB  0028   419          PUSHR   #^M<R1,R2,R3,R4,R6,R7,R8,R9,R10>  ; Save regs
                     002C   420          ;
                     002C   421          ;
                     002C   422          ;    Determine XWB$W_REMSIZ - it can be no larger than XWB$W_LOCSIZ.
                     002C   423          ;
                     002C   424          ;
       52    30 A5   D0  002C   425          MOVL    XWB$L_VCB(R5),R2                ; Get RCB
  40 A5      42 A5   B1  0030   426          CMPW    XWB$W_REMSIZ(R5),XWB$W_LOCSIZ(R5) ; Compare sizes.
                05   1B  0035   427          BLEQU   20$                             ; If LEQU then okay
  42 A5      40 A5   D0  0037   428          MOVL    XWB$W_LOCSIZ(R5),XWB$W_REMSIZ(R5) ; Use smaller for REMSIZ
                     003C   429 20$:         ;
                     003C   430          ;
                     003C   431          ;    Links are to a given path by having a non-zero XWB$W_PATH value.
                     003C   432          ;    Since the path selection is forced the link is 'non-adaptive' but
                     003C   433          ;    is much more efficient since it uses a faster interface an may
                     003C   434          ;    use a larger buffer than RCB$W_ECLSEGSIZ.
                     003C   435          ;
                     003C   436          ;    If this is a 'non-adaptive' link, then convert the permanent copy
                     003C   437          ;    of the route-header since it's format is path dependent.
                     003C   438          ;
                     003C   439          ;
     53    38 A5   9A  003C   440          MOVZBL  XWB$W_PATH(R5),R3               ; Get path index
                18   13  0040   441          BEQL    30$                             ; If EQL, path is not forced
51   0000'C5        D0  0042   442          MOVL    XWB$L_PTR_RTHD(R5),R1           ; Get route-header pointer
          FFB6'    30  0047   443          BSBW    QRL$SETUP_CHAN                  ; Setup QRL channel
        0D 50      E9  004A   444          BLBC    R0,30$                          ; If LBC, no channel
0000'C5    54      90  004D   445          MOVB    R4,XWB$B_ADJ_INX(R5)            ; Save the Adjacency index
0000'C5    51      D0  0052   446          MOVL    R1,XWB$L_PTR_RTHD(R5)           ; Get route-header pointer
      71   53      D0  0057   447          MOVL    R3,-(R1)                        ; Store the route-header size
                     005A   448 30$:         ;
                     005A   449          ;
                     005A   450          ;    Determine the number of receive and transmit buffers 'donated' by
                     005A   451          ;    the system.
                     005A   452          ;
                     005A   453          ;    Although this buffer 'donation' does not strictly adhere to VMS
                     005A   454          ;    conventions it is bounded (fixed number of buffers per link in the
                     005A   455          ;    worst case) and allows for a significant performance gain and much
                     005A   456          ;    simpler code.  In addition, it means that processes can use DECnet
                     005A   457          ;    efficiently even though they may have low BYTLM quota, and hence
                     005A   458          ;    BYTLM does not have to be set ridiculously high (and thus rendering
                     005A   459          ;    it just about useless).
                     005A   460          ;
                     005A   461          ;    The number of buffers donated for transmits is currently derived
```

```
                     005A   462          ;    from the NCP "Pipeline Quota" parameter -- hence this pool use
                     005A   463          ;    (or misuse) is controlled by the system manager since the maximum
                     005A   464          ;    pool used will be "Pipeline Quota" times "Maximum Links".
                     005A   465          ;
                     005A   466          ;    The number of buffers used for receives has not been made a
                     005A   467          ;    parameter for simplicity.  It is possible that, rather than making
                     005A   468          ;    it a parameter, it would be better to bound the buffer occupancy
                     005A   469          ;    time by periodically:
                     005A   470          ;
                     005A   471          ;            - deallocating the oldest Rcv CXB's queue to the LSB
                     005A   472          ;              (it hasn't been ACKed yet)
                     005A   473          ;            - using the Special Kernel AST to copy partial messages
                     005A   474          ;              attached to Rcv IRP's to the user buffer (this is now
                     005A   475          ;              done using an attached CXB count rather than a timer).
                     005A   476          ;
                     005A   477          ;       NOTE:   Perhaps limiting the number of CXB's moved to the IRP
                     005A   478          ;               could make the above schemes work better.  This means
                     005A   479          ;               that the AST code would have to start processing the
                     005A   480          ;               LSB list (RCV_IRP?) after emptying the IRP list.
                     005A   481          ;
                     005A   482          ;    Keep in mind that allowing flexible receive buffering may save
                     005A   483          ;    wasted datalink bandwidth and CPU cycles by reducing the number
                     005A   484          ;    of back-pressure messages and data segment retransmission.
                     005A   485          ;
                     005A   486          ;       NOTE:   It has been found that being able to buffer at least
                     005A   487          ;               one incoming segment per logical-link is essential for
                     005A   488          ;               for performance -- otherwise too many backpressure
                     005A   489          ;               messages need to be sent.  The segment, once buffered,
                     005A   490          ;               should not be ACK'd until the user issue's a receive or
                     005A   491          ;               until the same segment is retransmitted by the remote
                     005A   492          ;               end (in which case the link should be backpressured
                     005A   493          ;               until the use issues a receive).
                     005A   494          ;
                     005A   495          ;               For this reason, it is essential for the system to
                     005A   496          ;               donate at least one receive buffer per logical-link.
                     005A   497          ;
                     005A   498          ;
          58    D4   005A   499          CLRL    R8                                      ; Take no additional quota
                     005C   500                                                          ; until implemented
                     005C   501                                                          ;
                     005C   502          ASSUME  NSP$C_MAX_XPW  LE  254                  ; Make sure it can fit in a byte
                     005C   503                                                          ;
57    62 A2    9A   005C   504          MOVZBL  RCB$B_ECL_RFLW(R2),R7                   ; Get default max XMT CXB's
          02    12   0060   505          BNEQ    60$                                     ; If NEQ, okay
          57    96   0062   506          INCB    R7                                      ; Else, use 1 as a minimum
   28     57    91   0064   507 60$:     CMPB    R7,#NSP$C_MAX_XPW                       ; With bounds ?
          04    1B   0067   508          BLEQU   70$                                     ; If LEQU, okay
57    95 AF    9A   0069   509          MOVZBL  NSP$B_MAX_XPW,R7                        ; Else use maximum
58    92 AF    9A   006D   510 70$:     MOVZBL  NSP$B_MAX_RBF,R8                        ; Max unACK'd rcv CXB's allowed
                     0071   511          ;
                     0071   512          ;
                     0071   513          ;    Complete IO$_ACCESS IRP with success
                     0071   514          ;
                     0071   515          ;
   50    01    D0   0071   516          MOVL    S^#SS$_NORMAL,R0                        ; Setup IOSB image
51    42 A5    3C   0074   517          MOVZWL  XWB$W_REMSIZ(R5),R1                     ; IOSB second longword
      FF85'    30   0078   518          BSBW    NET$CMPL_ACC                            ; Complete the access QIO
```

```
                                007B   519          ;
                                007B   520          ;
                                007B   521          ;    Setup the DATA LSB.
                                007B   522          ;
                                007B   523          ;
    50    00A4 C5    9E         007B   524          MOVAB   XWB$T_DT(R5),R0                 ; Get DATA LSB address
                     5E  10     0080   525          BSBB    SETUP_LSB                       ; Init it
 2C A0    00D4 C5    9E         0082   526          MOVAB   XWB$T_LI(R5),LSB$L_CROSS(R0)    ; Setup cross channel pointer
       2B A0       20  90      0088   527          MOVB    #LSB$M_BOM,LSB$B_STS(R0)        ; Next seg should set 'BOM' flag
          51       03  90      008C   528          MOVB    #3,R1                           ; Default "xmt packet window"
          57       51  91      008F   529          CMPB    R1,R7                           ; Larger than X_CXBQUO ?
                   0B  1E      0092   530          BGEQU   80$                             ; If GEQU yes, shrink it
    51    57       03  87      0094   531          DIVB3   #3,R7,R1                        ; Else, enlarge packet window
          51       51  80      0098   532          ADDB    R1,R1                           ; Use two thirds CXBQUO
                   51  96      009B   533          INCB    R1                              ; Plus one (to avoid zero)
                   03  11      009D   534          BRB     90$                             ; Continue
          51       57  90      009F   535 80$:     MOVB    R7,R1                           ; Else shrink XPW value
       0C A0       51  90      00A2   536 90$:     MOVB    R1,LSB$B_X_PKTWND(R0)           ; Setup "xmt packet window"
0B A0     FF56     CF  90      00A6   537          MOVB    NSP$B_ADJ_XPW,LSB$B_X_ADJ(R0)   ; Init window adjust counter
       0E A0       57  90      00AC   538          MOVB    R7,LSB$B_X_CXBQUO(R0)           ; Setup max xmt CXB to allocate
       29 A0       58  90      00B0   539          MOVB    R8,LSB$B_R_CXBQUO(R0)           ; Init max rcv CXB's NSP may
                               00B4   540                                                  ; buffer before passing some to
                               00B4   541                                                  ; the session layer (user space)
                               00B4   542          ;
                               00B4   543          ;
                               00B4   544          ;    Setup the Link-Service/Interrupt LSB
                               00B4   545          ;
                               00B4   546          ;
    50    00D4 C5    9E         00B4   547          MOVAB   XWB$T_LI(R5),R0                 ; Get LI LSB address
                     25  10     00B9   548          BSBB    SETUP_LSB                       ; Init it
 2C A0    00A4 C5    9E         00BB   549          MOVAB   XWB$T_DT(R5),LSB$L_CROSS(R0)    ; Setup cross channel pointer
       2B A0       01  90      00C1   550          MOVB    #LSB$M_LI,LSB$B_STS(R0)         ; Mark it as LI subchannel
       0C A0       01  90      00C5   551          MOVB    #1,LSB$B_X_PKTWND(R0)           ; Init the "xmt packet window"
          0A A0    96  90      00C9   552          INCB    LSB$B_X_REQ(R0)                 ; NSP says that 1 Interrupt
          29 A0    96  90      00CC   553          INCB    LSB$B_R_CXBQUO(R0)              ;  message is implicitly
                               00CF   554                                                  ;  requested upon connect
                               00CF   555          ;
                               00CF   556          ;
                               00CF   557          ;    Finish setting up the XWB and return to the event processor to
                               00CF   558          ;    change state and re-process the same event again.
                               00CF   559          ;
                               00CF   560          ;
    50 A5     4C A5    B0      00CF   561          MOVW    XWB$W_TIM_INACT(R5),XWB$W_TIMER(R5) ; Reset timer
       0E A5     10    AA      00D4   562          BICW    #XWB$M_STS_CON,XWB$W_STS(R5)        ; XWB now in "RUN" format
                               00D8   563                                                  ;
          07DE   8F    BA      00D8   564          POPR    #^M<R1,R2,R3,R4,R6,R7,R8,R9,R10>    ; Restore regs
          50     01    D0      00DC   565          MOVL    #1,R0                               ;
                       05      00DF   566          RSB
                               00E0   567
                               00E0   568
                               00E0   569
                               00E0   570 SETUP_LSB:                                       ; Common LSB initialization
                     21    BB  00E0   571          PUSHR   #^M<R0,R5>                      ; Save regs
60   30   00   6E     00    2C  00E2   572          MOVC5   #0,(SP),#0,#LSB$S_LSB,(R0)      ; Fill LSB with zero's
                     21    BA  00E8   573          POPR    #^M<R0,R5>                      ; Restore regs
                       05      00EA   574          RSB                                     ; Done
                               00EB   575
```

```
                        00EB    577          .SBTTL  NET$ALTENTRY    - Driver alternate entry point
                        00EB    578 ;+
                        00EB    579 ;
                        00EB    580 ;  This routine is called by the Executive to pass an "internal" IRP to the
                        00EB    581 ;  driver.  "Internal" IRP's are those not built via QIO.  These IRPs are used
                        00EB    582 ;  by higher level software used to request I/O and should not be confused with
                        00EB    583 ;  the IRPs built and passed by the Transport layer to NSP.  The action here is
                        00EB    584 ;  to setup the IRP fields as if the packet had been processed by the FDT
                        00EB    585 ;  routines.
                        00EB    586 ;
                        00EB    587 ;
                        00EB    588 ;  INPUTS:      R5 = UCB address
                        00EB    589 ;                R3 = IRP address
                        00EB    590 ;
                        00EB    591 ;  OUTPUTS:     R5-R0 may be clobbered.
                        00EB    592 ;
                        00EB    593 ;
                        00EB    594 ;-
                        00EB    595 NET$ALTENTRY::                                      ; Accept an "internal" IRP
14 0C A3    1F    E1    00EB    596          BBC     #31,IRP$L_PID(R3),110$             ; If BC, not legal ALSTART IRP
     0FE8 8F    BB    00F0    597          PUSHR   #^M<R3,R5,R6,R7,R8,R9,R10,R11>     ; Save regs
                        00F4    598                                                     ;
   51    32 A3    3C    00F4    599          MOVZWL  IRP$W_BCNT(R3),R1                  ; Get message size
              5B    D4    00F8    600          CLRL    R11                                ; Say "can't go to IPL 2"
              5C    10    00FA    601          BSBB    ALT_ENTRY                          ; Dispatch on function type
                        00FC    602                                                     ;
     0FE8 8F    BA    00FC    603 100$:    POPR    #^M<R3,R5,R6,R7,R8,R9,R10,R11>     ; Restore regs
       04 50    E9    0100    604          BLBC    R0,120$                            ; If LBC, IRP was not consumed
              05    0103    605          RSB                                        ; Done
                        0104    606
     50    2C    3C    0104    607 110$:    MOVZWL  #SS$_ABORT,R0                      ; Indicate error
 38 A3     50    3C    0107    608 120$:    MOVZWL  R0,IRP$L_IOST1(R3)                 ; Setup error status
00000000'GF    17    010B    609          JMP     G^COM$POST                         ; Another packet for the heap
                        0111    610
                        0111    611
```

```
                        0111   613          .SBTTL  NET$FDT_RCV     - Process IO$_READxBLK  requests
                        0111   614          .SBTTL  NET$FDT_XMT     - Process IO$_WRITExBLK requests
                        0111   615 ;+
                        0111   616 ;
                        0111   617 ;  The user message is segmented into CXB buffers which are queued to the
                        0111   618 ;  DATA LSB.  These CXB's are to be passed to the Transport layer for
                        0111   619 ;  transmission at the appropriate time.
                        0111   620 ;
                        0111   621 ;
                        0111   622 ;  INPUTS:       AP      Pointer to the QIO P1 parameter
                        0111   623 ;                R11-R9  Scratch
                        0111   624 ;                R8      Must be saved/restored if return to Exec for next
                        0111   625 ;                        FDT routine
                        0111   626 ;                R7      I/O function code without modifiers
                        0111   627 ;                R6      CCB address
                        0111   628 ;                R5      UCB address
                        0111   629 ;                R4      PCB address
                        0111   630 ;                R3      IRP address
                        0111   631 ;                R2-R0   Scratch
                        0111   632 ;
                        0111   633 ;  OUTPUTS:      R5,R3   Preserved
                        0111   634 ;
                        0111   635 ;                All other regs are clobbered.
                        0111   636 ;
                        0111   637 ;-
                        0111   638 NET$FDT_RCV::                                ; Receiver FDT routine
      48 A3    D4       0111   639          CLRL    IRP$L_SES_BUF(R3)          ; QIO does not use this buffer
2A A3   2A    A8       0114   640          BISW    #IRP$M_COMPLX!-            ; QIO interface uses complex,
                        0118   641                  IRP$M_CHAINED!-            ; chained buffers
                        0118   642                  IRP$M_FUNC,IRP$W_STS(R3)   ; and mark as "read" function
50   72'AF    9E       0118   643          MOVAB   B^RCV_COMMON,R0            ; Setup initial action routine
        04    11       011C   644          BRB     RW_FDT                     ; Continue
                        011E   645                                             ;
                        011E   646 NET$FDT_XMT::                               ; Xmitter's FDT routine
50   96'AF    9E       011E   647          MOVAB   B^XMT_COMMON,R0            ; Setup initial action routine
                        0122   648                                             ;
        28    BB       0122   649 RW_FDT: PUSHR   #^M<R3,R5>                 ; Save reg's
                        0124   650                                             ;
   52   6C    D0       0124   651          MOVL    P1(AP),R2                  ; Get user VA
51   04 AC    3C       0127   652          MOVZWL  P2(AP),R1                  ; Get length of transfer
     2C A3    D4       012B   653          CLRL    IRP$L_SVAPTE(R3)           ; Say "no buffer"
     30 A3    B4       012E   654          CLRW    IRP$W_BOFF(R3)             ; No byte count quota taken yet
32 A3    51    D0       0131   655          MOVL    R1,IRP$L_BCNT(R3)          ; Setup byte count
   5B    01    D0       0135   656          MOVL    #1,R11                     ; Say "okay to go to IPL 2"
         60    16       0138   657          JSB     (R0)                       ; Dispatch
                        013A   658                                             ;
         28    BA       013A   659          POPR    #^M<R3,R5>                 ; Restore reg's
                        013C   660                                             ;
                        013C   661          SETIPL  #IPL$_ASTDEL               ; Restore IPL
      06 50    E9       013F   662          BLBC    R0,100$                    ; If LBC, error
00000000'GF    17       0142   663          JMP     G^EXE$QIORETURN            ; Return to user with success
                        0148   664                                             ;
 06 50    1F    E4       0148   665 100$:   BBSC    #31,R0,110$                ; If BS, return error via IOSB
00000000'GF    17       014C   666          JMP     G^EXE$ABORTIO              ; Abort the I/O
00000000'GF    17       0152   667 110$:   JMP     G^EXE$FINISHIOC            ; Return error via IOSB
                        0158   668
```

```
                                0158   670
                                0158   671                      .ENABL  LSB
                                0158   672
                                0158   673 ALT_ENTRY:                                          ; ALTSTART dispatching
35 2A A3    01    E1            0158   674                BBC     #IRP$V_FUNC,IRP$W_STS(R3),ALT_XMT; If BC, write function
                                015D   675                                                    ;
                                015D   676 ALT_RCV:                                          ; Receiver's ALTSTART routine
                                015D   677                ;
                                015D   678                ;
                                015D   679                ;    Setup buffer address (if any)
                                015D   680                ;
                                015D   681                ;
            52    D4            015D   682                CLRL    R2                          ; Assume no attach buffer
48 A3    2C A3    D0            015F   683                MOVL    IRP$L_SVAPTE(R3),IRP$L_SES_BUF(R3) ; Copy buffer address, if any
            0C    13            0164   684                BEQL    RCV_COMMON                  ; If EQL, none
23 2A A3    05    E0            0166   685                BBS     #IRP$V_CHAINED,IRP$W_STS(R3),200$; If chained, report error
         2C A3    D4            016B   686                CLRL    IRP$L_SVAPTE(R3)            ; Detach buffer
      52    48 B3    D0         016E   687                MOVL    @IRP$L_SES_BUF(R3),R2       ; Get pointer to data region
                                0172   688                                                    ;
                                0172   689 RCV_COMMON:                                       ; Common receive entry point
                                0172   690                                                    ;
            5E    10            0172   691                BSBB    XMT_RCV_CO                  ; Co-routine common processing
         07F5    30            0174   692                BSBW    NEW_RCV_IRP                 ; Queue the request
         1C A8    D5           0177   693                TSTL    LSB$L_R_IRP(R8)             ; IRP still there ?
            11    13            017A   694                BEQL    100$                        ; If EQL no, sent to IOPOST
                                017C   695                ;
                                017C   696                ;
                                017C   697                ;    If the receiver is back-pressured off, then open it up again.
                                017C   698                ;
                                017C   699                ;
1C A5    0800 8F    AA          017C   700                BICW    #XWB$M_FLG_TBPR,XWB$W_FLG(R5)   ; Assume toggling not needed
   06 0E A5    06    E1         0182   701                BBC     #XWB$V_STS_RBP,XWB$W_STS(R5),100$; If BC, not back-pressured off
1C A5    0800 8F    A8          0187   702                BISW    #XWB$M_FLG_TBPR,XWB$W_FLG(R5)   ; Toggle back-pressure
            05    05            018D   703 100$:          RSB                                 ; Done
                                018E   704
            50    2C    3C      018E   705 200$:          MOVZWL  #SS$_ABORT,R0               ; Setup error status
                        05      0191   706                RSB                                 ; Done
                                0192   707                      .DSABL  LSB
                                0192   708
```

```
                                                E 3
                     0192   710
                     0192   711 ALT_XMT:                                        ; Xmitter's ALTSTART routine
          52  2C B3  D0 0192   712             MOVL    @IRP$L_SVAPTE(R3),R2     ; Get start of data
                     0196   713                                                 ;
                     0196   714 XMT_COMMON:
    0D 20 A3   06    E1 0196   715             BBC     #IO$V_INTERRUPT,IRP$W_FUNC(R3),50$ ; If BC, DATA subchannel
                 2E  10 019B   716             BSBB    XMT_INT_CO               ; INT msg setup co-routine
    58  00D4 C5    9E 019D   717             MOVAB   XWB$T_LI(R5),R8          ; Get LS/INT LSB
      57   C6'AF    9E 01A2   718             MOVAB   B^100$,R7                ; Replace build routine ptr
                 07  11 01A6   719             BRB     60$                      ; Contine
                 28  10 01A8   720 50$:       BSBB    XMT_RCV_CO               ; Common setup co-routine
      57  0D36'CF    9E 01AA   721             MOVAB   W^XMT_COPY,R7            ; Setup message building routine
                     01AF   722 60$:          ;
                     01AF   723               ;
                     01AF   724               ;    Attach IRP to request list
                     01AF   725               ;
                     01AF   726               ;
        50    10 A8  9E 01AF   727             MOVAB   LSB$L_X_PND(R8),R0       ; Perpare for scan
           51  50    D0 01B3   728 70$:       MOVL    R0,R1                    ; Make a copy
           50  60    D0 01B6   729             MOVL    (R0),R0                  ; Get next IRP
                 F8  12 01B9   730             BNEQ    70$                      ; If NEQ, not last
           61  53    D0 01BB   731             MOVL    R3,(R1)                  ; Attach IRP to end of list.
                     01BE   732               ;
                     01BE   733               ;
                     01BE   734               ;    Start building and sending data segments
                     01BE   735               ;
                     01BE   736               ;
                 67  16 01BE   737             JSB     (R7)                     ; Build as many CXB's as we can
           52  58    D0 01C0   738             MOVL    R8,R2                    ; Setup LSB ptr for subr call
              0714   31 01C3   739             BRW     CALC_HXS_XMT             ; Determine new HXS value
                     01C6   740                                                 ; -- update XWB$M_FLG_WHGL
                     01C6   741
                     01C6   742
          8E    D5 01C6   743 100$:      TSTL    (SP)+
        06A1    31 01C8   744            BRW     CHK_INT_AVL_R8                ; Avoid call to CALC_HXS_XMT
                     01CB   745                                                 ; Schedule Interrupt message
                                                                               ; transmission if possilbe
```

```
                                01CB   747
                                01CB   748                      .ENABL  LSB
                                01CB   749 XMT_INT_CO:                                  ; Xmt INT message co-routine
                          4A    10 01CB   750                   BSBB    COPY_INT_DATA   ; Copy data to IRP
                       44 50    E9 01CD   751                   BLBC    R0,900$         ; If LBC, error
                          0C    11 01D0   752                   BRB     10$             ; Continue
                                01D2   753                                             ;
                                01D2   754 XMT_RCV_CO:                                  ; Common XMT/RCV co-routine
                    38 A3       01 B0 01D2   755                MOVW    #SS$_NORMAL,IRP$L_IOST1(R3) ; Init status
                    3A A3       51 B0 01D6   756                MOVW    R1,IRP$L_IOST1+2(R3) ; Save buffer size
                    3C A3       52 D0 01DA   757                MOVL    R2,IRP$L_IOST2(R3) ; Save buffer address
                                01DE   758                                             ;
                                01DE   759
                                01DE   760 10$:     ASSUME  IRP$L_IOQFL  EQ  0
                                01DE   761          ASSUME  IRP$B_QUO    EQ  4
                                01DE   762          ASSUME  IRP$B_CXBCNT EQ  5
                                01DE   763
                          63    7C 01DE   764                 CLRQ    (R3)              ; Clear linkage and CXB quota
                                01E0   765                 ;
                                01E0   766                 ;
                                01E0   767                 ;    Switch to XWB context, verify RUN state, locate LSB
                                01E0   768                 ;
                                01E0   769                 ;
                                01E0   770                 SETIPL  #NET$C_IPL          ; Synchronize with XWB, etc.
                 55   18 A3       01 CB 01E3   771          BICL3   #1,IRP$L_WIND(R3),R5 ; Go to XWB context
                             18    13 01E8   772             BEQL    200$              ; If EQL, no XWB
                          1E A5    05 91 01EA   773          CMPB    #XWB$C_STA_RUN,XWB$B_STA(R5) ; RUN state ?
                             1B    12 01EE   774             BNEQ    300$              ; If NEQ no, return error
                       52    30 A5 D0 01F0   775          MOVL    XWB$L_VCB(R5),R2    ; Setup RCB pointer
                    58    00A4 C5  9E 01F4   776          MOVAB   XWB$T_DT(R5),R8     ; Get DATA LSB
                                01F9   777                 ;
                                01F9   778                 ;
                                01F9   779                 ;    Call back
                                01F9   780                 ;
                                01F9   781                 ;
                             9E    16 01F9   782          JSB     @(SP)+
                                01FB   783                 ;
                                01FB   784                 ;
                                01FB   785                 ;    Send any new messages and return "success"
                                01FB   786                 ;
                                01FB   787                 ;
                          FE02'    30 01FB   788          BSBW    NET$SCH_MSG         ; Schedule message transmission
                       50   01    3C 01FE   789          MOVZWL  S^#SS$_NORMAL,R0    ; Indicate success
                                05 0201   790          RSB                         ; Done.
                                0202   791
              50    000000AC 8F    D0 0202   792 200$:   MOVL    #SS$_FILNOTACC,R0   ; Say "no logical-link"
                             09    11 0209   793          BRB     900$              ; Continue
                    50   20E4 8F    3C 020B   794 300$:   MOVZWL  #SS$_LINKABORT,R0  ; Say "not in RUN state"
                      00 50    1F    E2 0210   795 800$:   BBSS    #31,R0,900$       ; Say "return error via IOSB"
                             8E    D5 0214   796 900$:   TSTL    (SP)+             ; Pop co-routine address
                                05 0216   797          RSB                         ; Done
                                0217   798
                                0217   799                      .DSABL  LSB
                                0217   800
```

```
                                0217   802
                                0217   803
                                0217   804
                                0217   805 COPY_INT_DATA:
                                0217   806          ;
                                0217   807          ;
                                0217   808          ;    This is an INTERRUPT message.
                                0217   809          ;
                                0217   810          ;    Probe data if QIO interface.  Move the data into the IRP starting
                                0217   811          ;    at IRP$L_IOST1 with IRP$W_BCNT used to contain the size.
                                0217   812          ;
                                0217   813          ;    When the sequence number is assigned to this IRP (see NET$SCH_MSG)
                                0217   814          ;    the IO$V_INTERRUPT flag is cleared in IRP$W_FUNC in order to flag
                                0217   815          ;    the IRP state change.
                                0217   816          ;
                                0217   817          ;
          50   029C 8F    3C   0217   818          MOVZWL  #SS$_TOOMUCHDATA,R0                     ; Assume length violation
                  51    10    D1   021C   819          CMPL    #16,R1                                  ; Check data length
                        20    1F   021F   820          BLSSU   100$                                    ; If LSSU, too much data
             0F 0C A3   1F    E0   0221   821          BBS     #31,IRP$L_PID(R3),30$                   ; If BS, then ALSTART interface
       54    0B A3    02    00    EF   0226   822          EXTZV   #0,#2,IRP$B_RMOD(R3),R4                 ; Get mode for probe
                  50    0C    3C   022C   823          MOVZWL  #SS$_ACCVIO,R0                          ; Assume
                                0235   824          IFNORD  R1,(R2),100$,R4                         ; Goto 500$ if can't read data
                                0235   825          ;
                                0235   826          ;
                                0235   827 30$:     ASSUME  IRP$L_IOST1     EQ   0+IRP$L_MEDIA      ; Make sure there's
                                0235   828          ASSUME  IRP$L_IOST2     EQ   4+IRP$L_MEDIA      ; enough scratch space
                                0235   829          ASSUME  IRP$Q_STATION   EQ   8+IRP$L_MEDIA      ; for the data in the
                                0235   830                                                          ; IRP itself.
                                0235   831          ;
                        28    BB   0235   832          PUSHR   #^M<R3,R5>                              ; Save regs
    38 A3       62    51    28   0237   833          MOVC3   R1,(R2),IRP$L_IOST1(R3)                 ; Copy data into IRP
                        28    BA   023C   834          POPR    #^M<R3,R5>                              ; Restore regs
                                023E   835          ;
                  50    01    D0   023E   836          MOVL    #1,R0                                   ; Say "success"
                        05    05   0241   837 100$:    RSB                                             ; Done
                                0242   838          ;
                                0242   839          ;
```

```
                                                H 3
                                .SBTTL  NET$UNSOL_INTR  - Receive from Transport Layer
                        ;++
                        ;
                        ;  The following "unsolicited interrupt" routine is called by Transport
                        ;  whenever it has received a message addressed to NSP.  NSP must process the
                        ;  message completely and return to Transport without forking.  The message can
                        ;  be found in a single buffer of "complex chained" (CXB) format.  If NSP
                        ;  wishes to keep the message it must zero its the CXB pointer before returning
                        ;  to Transport.
                        ;
                        ;  NSP may need to return the message to its sender.  For instance, the
                        ;  message may be addressed to a link which no longer exists.  If this
                        ;  is the case, the CXB is kept and used as the context block for soliciting
                        ;  permission to transmit.
                        ;
                        ;  INPUTS:      R8      Scratch
                        ;               R7      Length of NSP message  (w/o route-header)
                        ;               R6      Address of CXB containing the message
                        ;               R5-R3   Scratch
                        ;               R2      RCB address
                        ;               R1      Pointer to first NSP byte in message
                        ;               R0      Scratch
                        ;
                        ;                       CXB$L_R_RCB     RCB address                     (copy of R2)
                        ;                       CXB$L_R_MSG     Ptr to 1st NSP byte in message  (copy of R1)
                        ;                       CXB$W_R_BCNT    Length of NSP message           (copy of R7)
                        ;                       CXB$W_R_SRCNOD  Source node address
                        ;                       CXB$W_R_DSTNOD  Local node address
                        ;                       CXB$B_R_FLG     LBS if CXB cannot be conusmed due to
                        ;                                       receiver buffering problems
                        ;                       CXB$W_R_PATH    Path number over which message was received
                        ;
                        ;  OUTPUTS: (upon return to Transport)
                        ;
                        ;                       R8,R7   Garbage
                        ;                       R6      0 if CXB was consumed.
                        ;                               Else, original CXB address with CXB$W_SIZE and
                        ;                                  CXB$B_TYPE unchanged.
                        ;                       R5-R0   Garbage
                        ;
                        ;--
                        NET$UNSOL_INTR::                                ; Receive message from Transport layer
0E00 8F     BB          0242        POPSHR  #^M<R9,R10,R11>             ; Extend to 'event' context -
                        0246                                            ;
     5B     D4          0246        CLRL    R11                         ; Say "can't go to IPL 2"
                        0248 ;;     BUMP    L,NDC+NDC$L_PRC(R5)         ; Inc 'packets rcvd' counter
     05     10          0248        BSBB    RCV_MSG                     ; Process received message
                        024A                                            ;
0E00 8F     BA          024A        POPR    #^M<R9,R10,R11>             ; Revert to 'fork' context
            05          024E        RSB                                 ; Return to Transport
                        024F
                        024F    RCV_MSG::
                        024F                ;
                        024F                ;   Map the message into an event code and check for message size
                        024F                ;   violations.
```

```
                        024F    898          ;
                        024F    899          ;
                        024F    900          ASSUME  RCVMAP_B_MSG    EQ  0
                        024F    901          ASSUME  RCVMAP_B_SIZ    EQ  1
                        024F    902          ASSUME  RCVMAP_B_EVT    EQ  2
                        024F    903          ASSUME  NSP$C_MSG_DATA  EQ  0
                        024F    904
        59    81    9A  024F    905          MOVZBL  (R1)+,R9                 ; Get message type code
     39 A6    59    90  0252    906 5$:      MOVB    R9,CXB$B_R_NSPTYP(R6)   ; Store it
  58    FDAB  CF    9E  0256    907          MOVAB   NET$AT_RCVMSG+1,R8      ; Assume DATA message
    59    9F  8F    93  025B    908          BITB    #^C<NSP$M_DATA_EOM!-    ;
                        025F    909                      NSP$M_DATA_BOM>,R9 ; Test all but DATA msg modifier bits
              0C    13  025F    910          BEQL    20$                     ; Br if DATA message
  88    FFFF  8F    B1  0261    911 10$:     CMPW    #RCVMAP_C_END,(R8)+     ; Advance to next entry
              12    13  0266    912          BEQL    35$                     ; If NEQ, at end of table -- ignore msg
        88    59    91  0268    913          CMPB    R9,(R8)+                ; Message type match ?
              F4    12  026B    914          BNEQ    10$                     ; If NEQ no, loop to try again
        50    88    98  026D    915 20$:     CVTBL   (R8)+,R0                ; Get min size by message type
              16    19  0270    916          BLSS    40$                     ; If LSS then variable message size
        57    50    C2  0272    917          SUBL    R0,R7                   ; Dec. msg length by expected size
              16    13  0275    918          BEQL    50$                     ; If EQL then okay
           00AA     31  0277    919 30$:     BRW     FMT_ERROR               ; Message too short
                        027A    920 35$:     ;
                        027A    921          ;
                        027A    922          ;    Message type unknown.  Check to see if its a retransmitted CI, and
                        027A    923          ;    if so, convert the code in R9 and try again.  Else, report the
                        027A    924          ;    event.
                        027A    925          ;
                        027A    926          ;
  68 8F     59    91  027A    927          CMPB    R9,#NSP$C_MSG_CR        ; Retransmitted Connect Initiate ?
              05    12  027E    928          BNEQ    37$                     ; If NEQ no, report event
        59    18    90  0280    929          MOVB    S^#NSP$C_MSG_CI,R9      ; Convert to CI
              CD    11  0283    930          BRB     5$                      ; Look up table entry for CI's
           009C     31  0285    931 37$:     BRW     UNK_MSG                 ; Message type unknown
                        0288    932 40$:     ;
                        0288    933          ;
                        0288    934          ;    Message type found, continue processing
                        0288    935          ;
                        0288    936          ;
        57    50    C0  0288    937          ADDL    R0,R7                   ; Reduce message by minimum size
              EA    19  028B    938          BLSS    30$                     ; If LSS then too short
        18    59    91  028D    939 50$:     CMPB    R9,S^#NSP$C_MSG_CI      ; Is message Connect Initiate ?
              59    12  0290    940          BNEQ    70$                     ; If NEQ, no
                        0292    941          ;
                        0292    942          ;
                        0292    943          ;    First check to see if this is a message being returned since
                        0292    944          ;    the remote node is unreachable.
                        0292    945          ;
                        0292    946          ;
1C 38 A6    01    E1  0292    947          BBC     #1,CXB$B_R_FLG(R6),60$  ; If BC then not "return to sender"
                        0297    948                                          ;; use a symbol
     53   02 A1    3C  0297    949          MOVZWL  2(R1),R3                ; Get local link i.d.
           FD62'    30  029B    950          BSBW    NET$XWB_LOCLNK          ; Switch to XWB context
                        029E    951                                          ; -- destroys R4
           0F 55    E8  029E    952          BLBS    R5,55$                  ; No associated XWB if LBS
           36 A6    B1  02A1    953          CMPW    CXB$W_R_SRCNOD(R6),-    ; Does the remote node i.d. match ?
           3A A5        02A4    954                  XWB$W_REMNOD(R5)        ;
```

```
                                                J 3
                08   12  02A6   955          BNEQ    55$                     ; If NEQ then no, forget it
        57  0000'8F   3C  02A8   956          MOVZWL  #NETEVT$_RTS,R7         ; Setup "returned to sender" event code
              0071   31  02AD   957          BRW     200$                    ; Dispatch to process the event
              0071   31  02B0   958 55$:      BRW     DISCARD
                         02B3   959 60$:      ;
                         02B3   960           ;
                         02B3   961           ;    Its an incoming connect request.  Create an XWB
                         02B3   962           ;
                         02B3   963           ;
                81   B5  02B3   964          TSTW    (R1)+                   ; Skip over local link field
                C0   12  02B5   965          BNEQ    30$                     ; Message format error if non-zero
        53   36  A6   3C  02B7   966          MOVZWL  CXB$W_R_SRCNOD(R6),R3   ; Get source node address for subr calls
             50  53   D0  02BB   967          MOVL    R3,R0                   ; Setup remote node address for subr.
               FD3F'   30  02BE   968          BSBW    TR$TEST_REACH           ; Is it reachable ?
              60  50   E9  02C1   969          BLBC    R0,DISCARD              ; If LBC, not reachable -- forget it
        55    14  A2   D0  02C4   970          MOVL    RCB$L_ACP_UCB(R2),R5    ; Get UCB address
               FD35'   30  02C8   971          BSBW    NET$CREATE_XWB          ; Get a new XWB and link slot
              57  55   E8  02CB   972          BLBS    R5,NO_RSRC              ; Br if no resources
                         02CE   973 ;;        BUMP    W,NDC$W_CRC(R10)        ;; need to account for resource errors
                         02CE   974           BUMP    W,NDC+NDC$W_CRC(R5)     ; Increment "connects received"
                32  A6   B0  02D9   975          MOVW    CXB$W_R_PATH(R6),-      ; Store the path over which the message
            0110  C5       02DC   976                  XWB$W_CI_PATH(R5)       ; was received
        54  A5    01   A3  02DF   977          SUBW3   #1,XWB$W_RETRAN(R5),-   ; Init PROGESS -- we want to break the
                52  A5       02E3   978                  XWB$W_PROGRESS(R5)      ; link if we timeout before user issues
                             02E5   979                                          ; the IO$_ACCESS function
        3C  A5    81   B0  02E5   980          MOVW    (R1)+,XWB$W_REMLNK(R5)  ; Store remote link address
                30   11  02E9   981          BRB     100$                    ; Continue in common
                         02EB   982 70$:      ;
                         02EB   983           ;
                         02EB   984           ;    Not a Connect Intiate.  Find the XWB and verify the partner's
                         02EB   985           ;    node address.
                         02EB   986           ;
                         02EB   987           ;
    53    81   3C  02EB   988          MOVZWL  (R1)+,R3                ; Get 'destination' link address
            FDOF'   30  02EE   989          BSBW    NET$XWB_LOCLNK          ; Switch to XWB context
                        02F1   990                                          ; -- destroys R4
          31  55   E8  02F1   991          BLBS    R5,RTS_NLT              ; No associated XWB if LBS
                        02F4   992 ;        TSTW    XWB$W_REMLNK(R5)        ; Remote link address known yet?
                        02F4   993 ;        BEQL    75$                     ; If EQL no, skip remote node check
                        02F4   994 ;                                        ; (for "cluster" node implementation
        36  A6    B1  02F4   995          CMPW    CXB$W_R_SRCNOD(R6),-    ; Is msg from proper remote?
        3A  A5        02F7   996                  XWB$W_REMNOD(R5)        ; node ?
                        02F9   997 ;        BEQL    80$                     ; If so, continue
                        02F9   998 ;        TSTW    XWB$W_REMNOD(R5)        ; Was the remote address zero?
              2A   12   02F9   999          BNEQ    RTS_NLT                 ; If not, branch
                        02FB  1000 75$: ;   MOVW    CXB$W_R_SRCNOD(R6),-    ; Else udpate to use new address
                        02FB  1001      ;           XWB$W_REMNOD(R5)        ; (this is the local node starting
                        02FB  1002                                          ;  and changing its address)
                        02FB  1003 80$:      ;
                        02FB  1004           ;
                        02FB  1005           ;    If the state is Connect Initiate Sending (CIS) or Connect Ack
                        02FB  1006           ;    Received (CAR) then no remote link address has been established
                        02FB  1007           ;    yet.
                        02FB  1008           ;
                        02FB  1009           ;    If the state is Closed (CLO) then the logical-link has been
                        02FB  1010           ;    dissolved at this end we must send a "No-link, terminate" message.
                        02FB  1011           ;
```

```
                                                K 3
                     02FB  1012                ;
                     02FB  1013                ASSUME  XWB$C_STA_CLO  EQ  0
                     02FB  1014                ASSUME  XWB$C_STA_CIS  EQ  1
                     02FB  1015                ASSUME  XWB$C_STA_CAR  EQ  2
                     02FB  1016
                     02FB  1017                BUMP    L,NDC+NDC$L_PRC(R5)       ; Inc 'packets rcvd' counter
02    1E A5    91    0306  1018                CMPB    XWB$B_STA(R5),#2          ; Have remote link address yet ?
         09    1A    030A  1019                BGTRU   90$                       ; If GTRU then yes
      1E A5    95    030C  1020                TSTB    XWB$B_STA(R5)             ; CLOSED state?
         14    13    030F  1021                BEQL    RTS_NLT                   ; If EQL yes, there's no link
3C A5    61    B0    0311  1022                MOVW    (R1)+,XWB$W_REMLNK(R5)    ; Store remote link address
3C A5    81    B1    0315  1023 90$:           CMPW    (R1)+,XWB$W_REMLNK(R5)    ; Is the remote link correct ?
         0A    12    0319  1024                BNEQ    RTS_NLT                   ; If not, branch
   52    57    D0    031B  1025 100$:          MOVL    R7,R2                     ; Setup # of unaccounted for msg bytes
   57    68    9A    031E  1026                MOVZBL  (R8),R7                   ; Get corresponding event code
      FCDC'    31    0321  1027 200$:          BRW     NET$EVENT                 ; Process the message
                     0324  1028
                     0324  1029 FMT_ERROR:                                      ; Message format error
                     0324  1030 UNK_MSG:                                        ; Unknown message type
                     0324  1031 DISCARD:                                        ; Discard message
                  05 0324  1032                RSB                           ; Ignore the message
                     0325  1033                                                ; (someday may want to log them)
                     0325  1034
```

```
                     0325  1036          .SBTTL  ACT$RTS_NLT     - Return to sender as "no-link-terminate"
                     0325  1037 ;+
                     0325  1038 ;
                     0325  1039 ;  The logical-link addressed by this message does not exist or could not be
                     0325  1040 ;  created.  Use the CXB as the context block with which to solicit Transport's
                     0325  1041 ;  permission to send a response message.
                     0325  1042 ;
                     0325  1043 ;  If the received message type was a Connect Initiate, then send a
                     0325  1044 ;  "no-resources" message.  Else, send a "no-link-terminate" message.
                     0325  1045 ;
                     0325  1046 ;
                     0325  1047 ;  INPUTS:  R6      CXB address
                     0325  1048 ;           R5      Not used
                     0325  1049 ;           R4-R0   Scratch
                     0325  1050 ;
                     0325  1051 ;  OUTPUTS: R6      CXB address or 0 if the CXB is consumed
                     0325  1052 ;           R4-R0   Garbage
                     0325  1053 ;
                     0325  1054 ;           All other regs are preserved
                     0325  1055 ;
                     0325  1056 ;-
                     0325  1057 RTS_NLT:                                 ; Return to sender as no-link-terminate
                     0325  1058 NO_RSRC:                                 ; No resources for inbound connect
                     0325  1059                                          ; fall thru to ACT$RTS_NLT
                     0325  1060 ACT$RTS_NLT::                            ; Return to sender as no-link-terminate
     07E0 8F    BB   0325  1061          PUSHR   #^M<R5,R6,R7,R8,R9,R10> ; Save regs
                     0329  1062                                          ;
 51    2C A6    D0   0329  1063          MOVL    CXB$L_R_MSG(R6),R1      ; Recover original rcv'd msg pointer
 61    48 8F    91   032D  1064          CMPB    #NSP$C_MSG_DC,(R1)      ; Is this a DC message
          12    13   0331  1065          BEQL    5$                      ; If EQL yes, ignore the message
     52   09    D0   0333  1066          MOVL    #9,R2                   ; Specify enough bytes to include NSP
                     0336  1067                                          ; header
        0AF0    30   0336  1068          BSBW    CLONE_RCV_CXB           ; Clone a copy of a rcv'd CXB
      09 50     E9   0339  1069          BLBC    R0,5$                   ; If LBC then failure
 2C A6    51    D0   033C  1070          MOVL    R1,CXB$L_R_MSG(R6)      ; Save message pointer
     55   56    D0   0340  1071          MOVL    R6,R5                   ; Use CXB as 'request block'
          05    10   0343  1072          BSBB    10$                     ; Continue in common
                     0345  1073                                          ;
     07E0 8F    BA   0345  1074 5$:      POPR    #^M<R5,R6,R7,R8,R9,R10> ; Save regs
                05    0349  1075          RSB                             ; Done
                     034A  1076
                     034A  1077
                     034A  1078 10$:     ;
                     034A  1079          ;
                     034A  1080          ;  Solicit permission from Transport to transmit a message.  Note
                     034A  1081          ;  that the request could suspend us indefinitely.  The call is
                     034A  1082          ;  made with:
                     034A  1083          ;
                     034A  1084          ;
                     034A  1085          ;     R5        Fork block address.
                     034A  1086          ;               The FPC,FR3,FR4 fields are all scratch and must not
                     034A  1087          ;               be modified by while Transport owns the fork block.
                     034A  1088          ;     R4        Destination node address
                     034A  1089          ;     R3        Index of LPD to xmit over
                     034A  1090          ;               Zero if Transport is to choose the LPD
                     034A  1091          ;     R1,R0     Scratch
                     034A  1092          ;
```

```
                     034A  1093          ;       (SP)    Return address of caller
                     034A  1094          ;     4(SP)     Return address of caller's caller
                     034A  1095          ;
                     034A  1096          ;
 54   36 A5    3C     034A  1097          MOVZWL  CXB$W_R_SRCNOD(R5),R4   ; Get remote node addresses
        53    D4     034E  1098          CLRL    R3                      ; Indicate no predetermined line
 52   28 A5    D0     0350  1099          MOVL    CXB$L_R_RCB(R5),R2      ; Setup RCB address
       FCA9'   30     0354  1100          BSBW    TR$SOLICIT              ; Solicit permission to xmit
                     0357  1101                                          ; (return may suspended indefinitely)
                     0357  1102          ;
                     0357  1103          ;
                     0357  1104          ;  Return from Transport with:
                     0357  1105          ;
                     0357  1106          ;       R7,R6   Scratch
                     0357  1107          ;       R5      Fork block address
                     0357  1108          ;       R4      Scratch
                     0357  1109          ;       R3      Not available -- must be saved/restored
                     0357  1110          ;       R2      RCB address
                     0357  1111          ;       R1      Scratch
                     0357  1112          ;       R0      Low bit set if permission granted
                     0357  1113          ;               Low bit clear if permission denied
                     0357  1114          ;
                     0357  1115          ;
   08 50    E8     0357  1116          BLBS    R0,20$                  ; If LBS then okay to xmit
 50   55    D0     035A  1117          MOVL    R5,R0                   ; Get block address
       FCA0'   30     035D  1118          BSBW    NET$DEALLOCATE          ; Deallocate the block
        56    11     0360  1119          BRB     40$                     ; Return
                     0362  1120 20$:     ;
                     0362  1121          ;
                     0362  1122          ;  Okay to xmit.  Build the message on the stack since the CXB will
                     0362  1123          ;  be corrupted before the final message is built.  Reformat the CXB,
                     0362  1124          ;  move the new message to the CXB, and return to Transport.
                     0362  1125          ;
                     0362  1126          ;  The message is built on the stack starting with the last byte so
                     0362  1127          ;  that when it is moved off of the stack conveniently.
                     0362  1128          ;
                     0362  1129          ;
 0E08 8F    BB     0362  1130          PUSHR   #^M<R3,R9,R10,R11>      ; Save regs
                     0366  1131                                          ;
        5B    D4     0366  1132          CLRL    R11                     ; Say "can't go to IPL 2"
                     0368  1133 ;;       BUMP    L,NDC$L_PSN(R10)        ; Update "packets sent"
                     0368  1134                                          ; Build the message backwards
 50   2C A5    D0     0368  1135          MOVL    CXB$L_R_MSG(R5),R0      ; Get ptr to original message
   7E    29    B0     036C  1136          MOVW    #NET$C_DR_NOLINK,-(SP)  ; Assume "no link terminate"
   60    18    91     036F  1137          CMPB    #NSP$C_MSG_CI,(R0)      ; If rcvd message is a Connect Initiate
         06    13     0372  1138          BEQL    25$                     ;  then terminate or...
 60   68 8F    91     0374  1139          CMPB    #NSP$C_MSG_CR,(R0)      ;  a retransmitted Connect Initiate
         03    12     0378  1140          BNEQ    30$                     ;
   6E    01    B0     037A  1141 25$:     MOVW    #NET$C_DR_RSU,(SP)      ;  then set reason as "no resources"
                     037D  1142 30$:     ;
                     037D  1143          ;
                     037D  1144          ;  Reverse destination and source of the logical link and
                     037D  1145          ;  node addresses in the new message
                     037D  1146          ;
                     037D  1147          ;
 7E   01 A0    B0     037D  1148          MOVW    NSP$W_DSTLNK(R0),-(SP)  ; Enter local link address as source
 7E   03 A0    B0     0381  1149          MOVW    NSP$W_SRCLNK(R0),-(SP)  ; Enter remote link address as dest
```

```
                7E    48 8F    90   0385  1150          MOVB    #NSP$C_MSG_DC,-(SP)     ; Enter msg type
                      7E       94   0389  1151          CLRB    -(SP)                   ; Enter the Transport "visits" field
                7E    34 A5    B0   038B  1152          MOVW    CXB$W_R_DSTNOD(R5),-(SP); Enter local node address
                7E    36 A5    B0   038F  1153          MOVW    CXB$W_R_SRCNOD(R5),-(SP); Enter remote node address
                51    48 A5    9E   0393  1154          MOVAB   CXB$T_X_XPORT(R5),R1    ; Get ptr to message to be built
                   53    51    D0   0397  1155          MOVL    R1,R3                   ; Make a working copy
                   83    02    90   039A  1156          MOVB    #TR3$C_MSG_DATA,(R3)+   ; Enter Transport message type
                   83    8E    D0   039D  1157          MOVL    (SP)+,(R3)+             ; Enter Dst,Src node addresses
                   83    8E    7D   03A0  1158          MOVQ    (SP)+,(R3)+             ; Enter Visits, NSP msg type, Dst and
                                    03A3  1159          ;                               ;     Src link addresses, Reason code
             57    53    51    C3   03A3  1160          SUBL3   R1,R3,R7                ; Setup message size
          52    00000000'GF    9E   03A7  1161          MOVAB   G^COM$DRVDEALMEM,R2     ; Address of I/O "end-action" routine
                   56    55    D0   03AE  1162          MOVL    R5,R6                   ; Setup buffer address
                   50    01    D0   03B1  1163          MOVL    #1,R0                   ; Tell Transport "okay to xmit"
                                    03B4  1164          ;
                         0E08 8F    BA   03B4  1165          POPR    #^M<R3,R9,R10,R11>      ; Restore regs
                                    03B8  1166          ;
                                    03B8  1167          ;
                                    03B8  1168          ;   Return to Transport with:
                                    03B8  1169          ;
                                    03B8  1170          ;    On return, the CXB and registers are setup as follows:
                                    03B8  1171          ;
                                    03B8  1172          ;
                                    03B8  1173          ;    +---------------+
                                    03B8  1174          ;    |   standard    |   11 bytes long.  CXB$L_FLINK and CXB$L_BLINK may
                                    03B8  1175          ;    |     VMS       |   be used by the Transport layer.  CXB$W_SIZE
                                    03B8  1176          ;    | buffer header |   must be correct.  CXB$B_TYPE must be DYN$C_CXB.
                                    03B8  1177          ;    +---------------+
                                    03B8  1178          ;    |               |   Starts with CXB$B_CODE (byte 11) and continues
                                    03B8  1179          ;    |     ECL       |   to CXB$C_LENGTH.  This area is read-only to
                                    03B8  1180          ;    |  pure area    |   Transport and below.  It cannot even be
                                    03B8  1181          ;    |               |   saved/restored.
                                    03B8  1182          ;    +---------------+
                                    03B8  1183          ;    |               |   Starts at CXB$C_LENGTH and is at least
                                    03B8  1184          ;    |   Datalink    |   CXB$C_DLL bytes long.  Used by the datalink for
                                    03B8  1185          ;    |     Layer     |   protocol header or state information.
                                    03B8  1186          ;    |  impure area  |
                                    03B8  1187          ;    +---------------+
                                    03B8  1188          ;    |               |   Must be quadword aligned and starting no sooner
                                    03B8  1189          ;    .     body of    .   than CXB$C_LENGTH + CXB$C_DLL (= CXB$C_HEADER)
                                    03B8  1190          ;    .     message   .
                                    03B8  1191          ;    |               |
                                    03B8  1192          ;    +---------------+
                                    03B8  1193          ;    |               |   Used by the datalink layer for protocol (e.g.,
                                    03B8  1194          ;    |   Datalink    |   checksum) or state information.  Must be at
                                    03B8  1195          ;    |     Layer     |   least CXB$C_TRAILER in length.
                                    03B8  1196          ;    |  impure area  |
                                    03B8  1197          ;    +---------------+
                                    03B8  1198          ;
                                    03B8  1199          ;        R7      Size of message
                                    03B8  1200          ;        R6      CXB address
                                    03B8  1201          ;        R5      Garbage
                                    03B8  1202          ;        R4      0 if "quick solicit" not requested
                                    03B8  1203          ;                Else, pointer to request block (XWB fork block) with
                                    03B8  1204          ;                FRK$L_FPC  pointing to the "quick solicit" routine
                                    03B8  1205          ;        R3      IRP address -- unmodified from call
                                    03B8  1206          ;        R2      Address of End-action routine to call on I/O competion
                                                        ;        R1      Ptr to 1st byte in standard Phase III route-header
```

```
                     03B8  1207          ;       R0      Low bit set   - if message is to be xmitted
                     03B8  1208          ;               Low bit clear - if no message to xmit.  In this case
                     03B8  1209          ;                               R7-R4,R2,R1 contain garbage.
                     03B8  1210          ;
                     03B8  1211          ;
54   D4              03B8  1212 40$:     CLRL    R4                      ; Say "quick solicit not wanted"
     05              03BA  1213          RSB
                     03BB  1214
```

```
                     03BB  1216          .SBTTL  ACT$RCV_CC      - Respond to a received Connect Confirm message
                     03BB  1217          .SBTTL  ACT$RCV_CA      - Respond to Connect Acknowledge
                     03BB  1218          .SBTTL  ACT$RCV_CI      - Process received Connect Initiate message
                     03BB  1219 ;++
                     03BB  1220 ;
                     03BB  1221 ;  These routines process received connect messages
                     03BB  1222 ;
                     03BB  1223 ;
                     03BB  1224 ;  INPUTS:  R8  Scratch
                     03BB  1225 ;           R7  Scratch
                     03BB  1226 ;           R6  CXB address
                     03BB  1227 ;           R5  XWB address
                     03BB  1228 ;           R4  Scratch
                     03BB  1229 ;           R3  Scratch
                     03BB  1230 ;           R2  Number of as yet unaccounted bytes in message
                     03BB  1231 ;           R1  Pointer to first unparsed byte in message
                     03BB  1232 ;           R0  Scratch
                     03BB  1233 ;
                     03BB  1234 ;  OUTPUTS: R8,R7,R4,R3,R2,R1 are garbage
                     03BB  1235 ;
                     03BB  1236 ;           R6  Preserved
                     03BB  1237 ;           R5  Preserved
                     03BB  1238 ;           R0  Standard VMS status code
                     03BB  1239 ;
                     03BB  1240 ;--
                     03BB  1241                  .ENABL  LSB
                     03BB  1242 ACT$RCV_CC::                             ; Repond to rcv'd CC msg
 50   2E   3C   03BB  1243          MOVZWL  #NET$C_DR_ZERO*2,R0         ; Assume error
   3C A5   B5   03BE  1244          TSTW    XWB$W_REMLNK(R5)            ; Test new remote link address
     2E    13   03C1  1245          BEQL    12$                         ; If EQL then illegal
    008D   30   03C3  1246          BSBW    PRS_CHR                     ; Parse link characteristics
  28 50    E9   03C6  1247          BLBC    R0,T2$                      ; If LBC then unsuccessful
  1E A5    91   03C9  1248          CMPB    XWB$B_STA(R5),-             ; Establish timer estimate
     02         03CC  1249                  #XWB$C_STA_CAR              ; unless its been done already
     02    13   03CD  1250          BEQL    10$
     23    10   03CF  1251          BSBB    ACT$RCV_CA                  ; Set timer seed value
                03D1  1252 10$:     UPDATE  L,R2,NDC+NDC$L_BRC(R5)      ; Bump "bytes received"
        58   31   03DD  1253          MOVZWL  #MSG$_CONFIRM,R8            ; Set mbx message code
          FC1D'   30   03E0  1254          BSBW    NET$SEND_CS_MBX             ; Notify user
        53 50    E9   03E3  1255          BLBC    R0,50$                      ; Br if error
          FC3F   30   03E6  1256          BSBW    NET$SETUP_RUN               ; Setup XWB for the RUN state
 57     0000'8F  3C   03E9  1257          MOVZWL  #NETEVT$_CC,R7              ; Set original event code
          FC0F'   31   03EE  1258          BRW     NET$COMPLEX_EV              ; Enter the RUN state and process new
                03F1  1259                                                  ; event
          0049   31   03F1  1260 12$:     BRW     100$                        ; Report protocol error
                03F4  1261
                03F4  1262 ACT$RCV_CA::                                     ; Respond to rcv'd CA msg
        53   01   B0   03F4  1263          MOVW    #1,R3                       ; Setup minimum timer value
   50    4A A5   B0   03F7  1264          MOVW    XWB$W_ELAPSE(R5),R0         ; Get elapsed time
          0B    13   03FB  1265          BEQL    15$                         ; If EQL then use minimum
        53   14   B0   03FD  1266          MOVW    #NSP$C_MAX_DELAY,R3         ; Setup maximum timer value
        53   50   B1   0400  1267          CMPW    R0,R3                       ; Compare max timer, elapsed time
          03    1E   0403  1268          BGEQU   15$                         ; If GEQU then R3 is smaller
        53   50   B0   0405  1269          MOVW    R0,R3                       ; Else use elapsed value
    4E A5    53   B0   0408  1270 15$:     MOVW    R3,XWB$W_DELAY(R5)          ; Setup seed value for timer
 50   010C C5   D0   040C  1271          MOVL    XWB$L_ICB(R5),R0            ; Get the ICB
50    04 A0    01   A1   0411  1272          ADDW3   #1,ICB$W_TIM_OCON(R0),R0; Get outbound connect timer (the 1
```

```
                                0416  1273                                    ; is for possible clock skew)
50 A5   50    50 A5    A3     0416  1274          SUBW3   XWB$W_TIMER(R5),R0,-     ; Replace TIMER with the amount of time
                              041C  1275                  XWB$W_TIMER(R5)         ; left before the connect times out
                 04    14     041C  1276          BGTR    17$                     ; If GTR then okay
           50 A5   01    B0     041E  1277          MOVW    #1,XWB$W_TIMER(R5)      ; Else use 1 second as minimum value
                         05     0422  1278 17$:     RSB                             ; May change state on return
                              0423  1279
                              0423  1280 ACT$RCV_CR::                             ; Process retransmitted CI message
                0100 8F    A8     0423  1281          BISW    #XWB$M_FLG_SCD,-        ; Set "send connect/disconnect" flag
                  1C A5             0427  1282                  XWB$W_FLG(R5)           ;
                         05     0429  1283          RSB                             ; Done
                              042A  1284
                              042A  1285 ACT$RCV_CI::                             ; Process CI message
                  0026   30     042A  1286          BSBW    PRS_CHR                 ; Parse the link characteristics
                OD 50    E9     042D  1287          BLBC    R0,100$                 ; Br on error
                  0080   30     0430  1288          BSBW    GETCTL                  ; Parse remainder of message
                07 50    E9     0433  1289          BLBC    R0,100$                 ; Br on error
                  FBC7'  31     0436  1290          BRW     NET$QUE_XWB             ; Queue XWB to NETACP
                              0439  1291
                              0439  1292 ;
                              0439  1293 ;
                              0439  1294 ;  Catch errors here.  In each case the return is with the low bit clear
                              0439  1295 ;  to flag a new event to be processed.
                              0439  1296 ;
                              0439  1297 ;
                    50    D5     0439  1298 50$:     TSTL    R0                      ; 0=> counted str format error
                    0B    12     043B  1299          BNEQ    200$                    ; If eql then protocol error
             57   00'8F    9A     043D  1300 100$:    MOVZBL  #NETEVT$_PROERR,R7      ; Setup new event
       46 A5   50    02    A7     0441  1301          DIVW3   #2,R0,XWB$W_X_REASON(R5); Set disconnect reason
                    08    11     0446  1302          BRB     300$                    ; Continue
                    01    B0     0448  1303 200$:    MOVW    #NET$C_DR_RSU,-         ; Setup disconnect reason as
                 46 A5             044A  1304                  XWB$W_X_REASON(R5)      ; "no resources"
              57   00'8F    9A     044C  1305          MOVZBL  #NETEVT$_MBXERR,R7      ; Setup new event
                  FBAD'  31     0450  1306 300$:    BRW     NET$PRE_EMPT            ; Pre-empt with new event
                              0453  1307
                              0453  1308
                              0453  1309                  .DSABL  LSB
                              0453  1310
```

```
                                0453  1312          .SBTTL  PRS_CHR          - Get characteristics from Connect message
                                0453  1313 ;
                                0453  1314 ;
                                0453  1315 ;
                                0453  1316 PRS_CHR:                                ; Get link characteristics
                                0453  1317          ;
                                0453  1318          ;
                                0453  1319          ;    If any part of the SERVICES field is not recognized then
                                0453  1320          ;    reject the connect.
                                0453  1321          ;
                                0453  1322          ;
     50    61    0C 8F    8B   0453  1323          BICB3   #^C<NSP$M_SRV_REQ>,(R1),R0 ; Get pertinent service bits
           50    01 91    12   0458  1324          CMPB    #NSP$C_SRV_REQ,R0         ; Are they correct ?
                 52 12    90   045B  1325          BNEQ    200$                      ; If NEQ no
           50    81 90    EF   045D  1326          MOVB    (R1)+,R0                  ; Get SERVICES field
                 02 EF    0460  1327          EXTZV   #NSP$V_SRV_FLW,-          ; Get flow control bits
      50   50    02       0462  1328                  #NSP$S_SRV_FLW,R0,R0      ;
           54    01 D0    0465  1329          MOVL    #XWB$M_PRO_NFC,R4         ; Assume "no-flow"
           50    00 91    0468  1330          CMPB    #NSP$C_SRV_NFC,R0         ; Is it ?
                 0F 13    046B  1331          BEQL    10$                       ; If EQL yes
           54    02 D0    046D  1332          MOVL    #XWB$M_PRO_SFC,R4         ; Assume "seg flow"
           50    01 91    0470  1333          CMPB    #NSP$C_SRV_SFC,R0         ; Is it ?
                 07 13    0473  1334          BEQL    10$                       ; If EQL yes
                 54 D4    0475  1335          CLRL    R4                        ; Assume "msg-flow"
           50    02 91    0477  1336          CMPB    #NSP$C_SRV_MFC,R0         ; Is it ?
                 33 12    047A  1337          BNEQ    200$                      ; If NEQ no, reject message
        5A A5    54 88    047C  1338 10$:     BISB    R4,XWB$B_PRO(R5)          ; Insert flow control info
                          0480  1339          ;
                          0480  1340          ;
                          0480  1341          ;    Parse the INFO field.  Ignore any part of the field which is
                          0480  1342          ;    not recognized.
                          0480  1343          ;
  50    81    FC 8F    8B   0480  1344          BICB3   #^C<NSP$M_INF_VER>,(R1)+,R0 ; Get NSP version, advance msg ptr
                          0485  1345
           54    04 D0    0485  1346          MOVL    #XWB$M_PRO_PH2,R4         ; Assume Phase II
           50    01 91    0488  1347          CMPB    #NSP$C_INF_V31,R0         ; Phase II ?
                 0F 13    048B  1348          BEQL    50$                       ; If EQL, no further capabilities
                 54 D4    048D  1349          CLRL    R4                        ; Init capabilities mask
           50    00 91    048F  1350          CMPB    #NSP$C_INF_V32,R0         ; Version 3.2 ?
                 08 13    0492  1351          BEQL    50$                       ; If EQL, no cross channel ACKing
           50    02 91    0494  1352          CMPB    #NSP$C_INF_V40,R0         ; Version 4.0 ?
                 03 12    0497  1353          BNEQ    50$                       ; If NEQ, version is unknown
           54    18 88    0499  1354          BISB    #XWB$M_PRO_CCA!-          ; Cross channel ACKing allowed
                          049C  1355                    XWB$M_PRO_NAR,R4        ; "No ACK requested" flag allowed
        5A A5    54 88    049C  1356 50$:     BISB    R4,XWB$B_PRO(R5)          ; Remember capabilities
                          04A0  1357          ;
                          04A0  1358          ;
                          04A0  1359          ;    Parse the SEGSIZ field
                          04A0  1360          ;
                          04A0  1361          ;    Make sure it's nonzero (the NSP spec has a higher minimum, but
                          04A0  1362          ;    TOPS 20 violates it and there's no sense being overly picky).
                          04A0  1363          ;
     50    004A 8F    3C   04A0  1364          MOVZWL  #<NET$C_DR_SEGSIZ>@1,R0 ; Assume illegal segment size
        42 A5    81 B0    04A5  1365          MOVW    (R1)+,XWB$W_REMSIZ(R5)    ; Get remote's rcv seg size
                 07 13    04A9  1366          BEQL    210$                      ; If EQL then illegal
           50    01 90    04AB  1367          MOVB    #1,R0                     ; Indicate success
                          05   04AE  1368          RSB
```

```
                04AF  1369
50  0E  D0  04AF  1370 200$:    MOVL    #NET$C_DR_PROTCL@1,R0   ; Indicate error
            05  04B2  1371 210$:    RSB                             ; Return with error
                04B3  1372
```

```
                                04B3  1374                    .ENABL  LSB
                                04B3  1375 ;
                                04B3  1376 ;
                                04B3  1377 ;  Supporting parse CI routines
                                04B3  1378 ;
                                04B3  1379 ;
 53   00A4 C5    9E  04B3  1380 GETCTL: MOVAB   XWB$B_LPRNAM(R5),R3     ; Setup destination pointer
        0098    30  04B8  1381         BSBW    MOVPRNAM                ; Move the dest. process name
                    04BB  1382                                         ;  - no return if error
 53   00B8 C5    9E  04BB  1383         MOVAB   XWB$B_RPRNAM(R5),R3     ; Setup destination pointer
        0090    30  04C0  1384         BSBW    MOVPRNAM                ; Move the src. process name
                    04C3  1385                                         ;  - no return if error
      54   81    90  04C3  1386         MOVB    (R1)+,R4                ; Save flags
      58   83    9E  04C6  1387         MOVAB   (R3)+,R8                ; Save current output ptr
                    04C9  1388                                         ; and advance past count field
        16 54    E9  04C9  1389         BLBC    R4,70$                  ; Br if no accounting info
      68   3D    90  04CC  1390         MOVB    #XWB$C_LOGIN-3,(R8)     ; Setup total space available
           6C    10  04CF  1391         BSBB    MOVCS_39                ; Move User field
                    04D1  1392                                         ;  - no return if error
           6A    10  04D1  1393         BSBB    MOVCS_39                ; Move Password field
           68    10  04D3  1394         BSBB    MOVCS_39                ; Move Account field
      53   58    C2  04D5  1395         SUBL    R8,R3                   ; Get count of bytes moved -
           53    D7  04D8  1396         DECL    R3                      ;  adjusting for count field
      68   53    90  04DA  1397         MOVB    R3,(R8)                 ; Store count
      52   53    C2  04DD  1398         SUBL    R3,R2                   ; Account for "optional" bytes
           03    11  04E0  1399         BRB     90$                     ; Continue
      68   03    D0  04E2  1400 70$:    MOVL    #3,(R8)                 ; A null access string is a 3
                    04E5  1401                                         ; (string count) followed by
                    04E5  1402                                         ; 3 zero's (substring counts)
 53     5B A5    3E  04E5  1403 90$:    MOVAW   XWB$B_DATA(R5),R3       ; Get next XWB field address
           63    D4  04E9  1404         CLRL    (R3)                    ; Assume no optional data
      50   01    D0  04EB  1405         MOVL    #1,R0                   ; Assume success
   0F 54   01    E1  04EE  1406         BBC     #1,R4,100$              ; Br if no optional data
                    04F2  1407         UPDATE  L,R2,NDC+NDC$L_BRC(R5)  ; Bump "bytes received"
        0088    30  04FE  1408         BSBW    MOVCSFX_17              ; Move optional data field
                    0501  1409                                         ;  NO return if error
           52    D5  0501  1410 100$:   TSTL    R2                      ; Any bytes left in message ?
           48    12  0503  1411         BNEQ    130$                    ; Illegal message if NEQ
                    0505  1412         ;
                    0505  1413         ;
                    0505  1414         ;  Move remote user i.d. to non-muliplexed portion of the XWB.
                    0505  1415         ;
                    0505  1416         ;  The RPRNAM field, as stored, is one of:
                    0505  1417         ;
                    0505  1418         ; <1 byte count><1 byte = 0><1 byte object type  not = 0>
                    0505  1419         ; <1 byte count><1 byte = 1><1 byte = 0><1-16 process name>
                    0505  1420         ; <1 byte count><1 byte = 2><1 byte = 0><4 byte UIC><1-12 process name>
                    0505  1421         ;
                    0505  1422         ;
           36    BB  0505  1423         PUSHR   #^M<R1,R2,R4,R5>        ; Save regs
                    0507  1424         ;
 51   00B8 C5    9E  0507  1425         MOVAB   XWB$B_RPRNAM(R5),R1     ; Point to remote process name
   53     6F A5    9E  050C  1426         MOVAB   XWB$B_RID(R5),R3        ; Point to permanent storage
 52   81    02    83  0510  1427         SUBB3   #2,(R1)+,R2             ; Get total number of bytes minus those
                    0514  1428                                         ; used for format type and object number
           21    15  0514  1429         BLEQ    110$                    ; If LEQ then no username text
      50   81    9A  0516  1430         MOVZBL  (R1)+,R0                ; Get format type
```

```
                     1C  13  0519  1431          BEQL    110$                    ; If EQL then no name text
                     50  D7  051B  1432          DECL    R0                      ; Format type 1?
                     07  13  051D  1433          BEQL    105$                    ; If EQL then yes
                     50  D7  051F  1434          DECL    R0                      ; Format type 2?
                     14  12  0521  1435          BNEQ    110$                    ; If NEQ then no, unknown format type
                 51  04  C0  0523  1436          ADDL    #4,R1                   ; Account for UIC bytes
                     51  D6  0526  1437  105$:    INCL    R1                      ; Skip over object number
                 52  81  9A  0528  1438          MOVZBL  (R1)+,R2                ; Get count field
                 52  10  91  052B  1439          CMPB    #XWB$C_RID,R2           ; Too large?
                     07  1F  052E  1440          BLSSU   110$                    ; If LSSU then use
                 83  52  90  0530  1441          MOVB    R2,(R3)+                ; Enter count field
             63  61  52  28  0533  1442          MOVC3   R2,(R1),(R3)            ; Enter remainder of text
                         0537  1443                                              ;
                     36  BA  0537  1444  110$:    POPR    #^M<R1,R2,R4,R5>        ; Restore regs
                 50  01  D0  0539  1445          MOVL    S^#SS$_NORMAL,R0        ; Indicate success
                     05  053C  1446          RSB                             ; Done
                         053D  1447
                         053D  1448
                         053D  1449
                         053D  1450  MOVCS_39:
                 61  27  91  053D  1451          CMPB    #39,(R1)                ; Test max string count value
                     09  1F  0540  1452          BLSSU   120$                    ; Br if too large
                 68  61  82  0542  1453          SUBB    (R1),(R8)               ; Account for space to be used
                     04  19  0545  1454          BLSS    120$                    ; Br if not enough space left
                    FAB6' 30  0547  1455          BSBW    NET$MOV_CSTR            ; Else move it
                     05  054A  1456          RSB                             ; Done
                         054B  1457
                     8E  D5  054B  1458  120$:    TSTL    (SP)+                   ; Pop return address
             50  0044 8F  3C  054D  1459  130$:    MOVZWL  #<NET$C_DR_ACCESS>*2,R0 ; Setup failure code
                     05  0552  1460          RSB
                         0553  1461
                         0553  1462                  .DSABL  LSB
```

```
                                                I  4

0553  1464
0553  1465 ;++
0553  1466 ;  MOVCSFX       Move counted string to fixed length field
0553  1467 ;  MOVCSFX_17    Move counted string to field 17 bytes long
0553  1468 ;  MOVPRNAM      Move process name
0553  1469 ;
0553  1470 ;
0553  1471 ;  The field pointed to by R1 is moved to the fixed length field pointed to by
0553  1472 ;  R3.  The resultant field is always stored as a counted string.  If the an
0553  1473 ;  error is encountered, the caller's  return address is poped off the stack
0553  1474 ;  and the return is to the caller's caller.
0553  1475 ;
0553  1476 ;  The process name field, as stored, is one of:
0553  1477 ;
0553  1478 ;  <1 byte count><1 byte = 0><1 byte object type  not = 0>
0553  1479 ;  <1 byte count><1 byte = 1><1 byte = 0><I-16 process name>
0553  1480 ;  <1 byte count><1 byte = 2><1 byte = 0><4 byte UIC><I-12 process name>
0553  1481 ;
0553  1482 ;  The process name source field is the same but without the count field
0553  1483 ;
0553  1484 ;
0553  1485 ;  INPUTS:      R3      Pointer to the desination field
0553  1486 ;               R2      Number of message bytes not yet accounted for
0553  1487 ;               R1      Pointer to first byte of message
0553  1488 ;               R0      Size of destination field (MOVCXFX only)
0553  1489 ;
0553  1490 ;             (SP)      Return address of caller
0553  1491 ;            4(SP)      Return address of caller's caller
0553  1492 ;
0553  1493 ;  OUTPUTS:     R3      Pointer to first byte beyond fixed length dest. field
0553  1494 ;               R2      Reduced by number of source field bytes moved
0553  1495 ;               R1      Pointer to first unmoved byte in message
0553  1496 ;               R0      1 if successfull, 0 otherwise
0553  1497 ;
0553  1498 ;--
0553  1499
0553  1500 ;+
0553  1501 ;    NOTE: These routines assume that the count in R2 of the remaining
0553  1502 ;          bytes left is a longword and that the total bytes to be moved
0553  1503 ;          is less than 255.
0553  1504 ;-
0553  1505
0553  1506                      .ENABL  LSB
0553  1507
0553  1508 MOVPRNAM:                                          ; Move process name
      34   BB   0553  1509           PUSHR   #^M<R2,R4,R5>    ; Save R4,R5; COPY R2
      0A   DD   0555  1510           PUSHL   #<NET$C_DR_FMT>*2  ; Assume error
                0557  1511                                   ;
52    02   D0   0557  1512           MOVL    #2,R2            ; Establish min. src field size
                055A  1513           $DISPATCH  TYPE=B,(R1),-  ; Dispatch on format type
                055A  1514           <-
                055A  1515               <0,  5$>,-           ;  non-zero object number only
                055A  1516               <1, 10$>,-           ;  object #0, I-16  taskname
                055A  1517               <2, 20$>,-           ;  object #0, 4 byte UIC, I-12 taskname
                055A  1518           >
      45   11   0564  1519           BRB     100$             ; Error if unrecognized format type
   01 A1   95   0566  1520 5$:       TSTB    1(R1)            ; Test object number
```

```
                                                J 4
                     40   13  0569  1521          BEQL    100$                    ; If EQL then format error
                     14   11  056B  1522          BRB     30$                     ; Continue in common
                     52   96  056D  1523 10$:      INCB    R2                      ; Inc for string count subfield
               52 02 A1   80  056F  1524          ADDB    2(R1),R2                ; Inc for string text subfield
                     07   11  0573  1525          BRB     25$                     ; Continue in common
                  52 05   80  0575  1526 20$:      ADDB    #4+1,R2                 ; Inc for UIC, string count subfields
               52 06 A1   80  0578  1527          ADDB    6(R1),R2                ; Inc for string text subfield
                  01 A1   95  057C  1528 25$:      TSTB    1(R1)                   ; Object number zero ?
                     2A   12  057F  1529          BNEQ    100$                    ; If NEQ then format error
                  83 52   90  0581  1530 30$:      MOVB    R2,(R3)+                ; Insert length of process name
                  50 13   D0  0584  1531          MOVL    #XWB$C_LPRNAM-1,R0      ; Establish fixed dst field size
                              0587  1532                                          ;  (not including the XWB$B_xPRNAM
                              0587  1533                                          ;   field which was just moved)
                     0C   11  0587  1534          BRB     40$                     ; Continue
                              0589  1535
                              0589  1536 MOVCSFX_17:
                  50 11   D0  0589  1537          MOVL    #17,R0                  ; Setup size of fixed field
                              058C  1538 MOVCSFX:
                     34   BB  058C  1539          PUSHR   #^M<R2,R4,R5>           ; Save R4,R5; COPY R2
                     7E   D4  058E  1540          CLRL    -(SP)                   ; Push error flag
                              0590  1541                                          ;
                  52 61   9A  0590  1542          MOVZBL  (R1),R2                 ; Get length of source string
                  52      D6  0593  1543          INCL    R2                      ; Update for count field
               04 AE 52   C2  0595  1544 40$:      SUBL    R2,4(SP)                ; Update remaining byte count
                     10   19  0599  1545          BLSS    100$                    ; Error if LSS
                  50 52   B1  059B  1546          CMPW    R2,R0                   ; Is source larger then destination
                     0B   1A  059E  1547          BGTRU   100$                    ; If GTRU then source was too long
63 50 00 61           52   2C  05A0  1548          MOVC5   R2,(R1),#0,R0,(R3)      ; Move the string, update R1,R3
                              05A6  1549                                          ;
                     35   BA  05A6  1550          POPR    #^M<R0,R2,R4,R5>        ; Restore R4,R2, UPDATE R2
                     50   96  05A8  1551          INCB    R0                      ; Flag success
                          05  05AA  1552          RSB
                              05AB  1553
                              05AB  1554 ;
                              05AB  1555 ;
                              05AB  1556 ;  Catch the errors here
                              05AB  1557 ;
                              05AB  1558 ;
                  35   BA      05AB  1559 100$:    POPR    #^M<R0,R2,R4,R5>        ; Restore R4,R2, UPDATE R2
                              05AD  1560                                          ; error code in R0
                  8E   D5     05AD  1561          TSTL    (SP)+                   ; Pop caller's return address
                       05     05AF  1562          RSB                             ; Return to its caller
                              05B0  1563
                              05B0  1564                  .DSABL  LSB
                              05B0  1565
```

```
                     05B0  1567          .SBTTL  ACT$RCV_RTS     - Receive CI message being "returned to sender"
                     05B0  1568          .SBTTL  ACT$RCV_Dx      - Recieve DI/DC message
                     05B0  1569          .SBTTL  ACT$ABORT       - Disconnect or abort a link
                     05B0  1570          .SBTTL  ACT$CANLNK      - Disconnect link due to user's $CANCEL
                     05B0  1571 ;+
                     05B0  1572 ;
                     05B0  1573 ;  The user is notified of the disconnect via the mailbox associated
                     05B0  1574 ;  with the link's UCB.  All pending IRPs are completed.
                     05B0  1575 ;
                     05B0  1576 ;
                     05B0  1577 ;  INPUTS:  R8  Scratch
                     05B0  1578 ;           R7  Scratch
                     05B0  1579 ;           R6  CXB address
                     05B0  1580 ;           R5  XWB address
                     05B0  1581 ;           R4  Scratch
                     05B0  1582 ;           R3  Scratch
                     05B0  1583 ;           R2  Number of as yet unaccounted bytes in message
                     05B0  1584 ;           R1  Pointer to first unparsed byte in message
                     05B0  1585 ;           R0  Scratch
                     05B0  1586 ;
                     05B0  1587 ;           Note:    R1,R2,R6 are listed above for ACT$RCV_Dx_xxx only.
                     05B0  1588 ;                    They carry no useful information for the other routines.
                     05B0  1589 ;
                     05B0  1590 ;  OUTPUTS: R6  Preserved
                     05B0  1591 ;           R5  Preserved
                     05B0  1592 ;
                     05B0  1593 ;           R8,R7,R4,R3,R2,R1,R0 are garbage
                     05B0  1594 ;
                     05B0  1595 ;--
                     05B0  1596 ACT$CANLNK::                                 ; Cancel I/O request
52   01   CE           05B0  1597          MNEGL   #1,R2                      ; No disconnect mailbox message
     06   11           05B3  1598          BRB     SET_X                      ; Continue
                       05B5  1599                                             ;
                       05B5  1600 ACT$RCV_RTS::                               ; Receive "returned to sender" CI msg
          27   3C       05B5  1601          MOVZWL #NET$C_DR_NOPATH,-          ; Phony up a received disconnect code
        44 A5           05B7  1602                  XWB$W_R_REASON(R5)         ; and fall thru to ACT$ABORT
                       05B9  1603                                             ;
                       05B9  1604 ACT$ABORT::                                 ; Abort a logical link
          52   D4       05B9  1605          CLRL    R2                         ; No disconnect data
   57   44 A5   3C       05BB  1606 SET_X:  MOVZWL   XWB$W_R_REASON(R5),R7     ; Received reason already set?
57   0000'8F   B1       05BF  1607          CMPW    #NET$C_DR_INVALID,R7       ;
          07   12       05C4  1608          BNEQ    10$                        ; If NEQ yes
       57   09   3C       05C6  1609          MOVZWL  #NET$C_DR_ABORT,R7         ; Else, use abort disconnect
     46 A5   57   B0       05C9  1610          MOVW    R7,XWB$W_X_REASON(R5)      ; Set disconnect reason for remote
          07   11       05CD  1611 10$:     BRB     CMPL_DISCON                ;
                       05CF  1612
                       05CF  1613
                       05CF  1614 ACT$RCV_Dx::
       57   81   3C       05CF  1615          MOVZWL  (R1)+,R7                   ; Get actual disconnect reason
          2A   B0       05D2  1616          MOVW    #NET$C_DR_CONF,-           ; Confirm DI message for remote
        46 A5           05D4  1617                  XWB$W_X_REASON(R5)         ;
                       05D6  1618
                       05D6  1619 CMPL_DISCON:                                ; Complete disconnect
                       05D6  1620          ;
                       05D6  1621          ;
                       05D6  1622          ;    Find disconnect reason code map table entry
                       05D6  1623          ;
```

```
                        05D6  1624          ;
44 A5    57    B0       05D6  1625          MOVW    R7,XWB$W_R_REASON(R5)   ; Setup reason code
      FA23'    30       05DA  1626          BSBW    NET$MAP_R_REASON        ; Find mapping table entry address
                        05DD  1627          ;
                        05DD  1628          ;
                        05DD  1629          ;    Notify user of disconnect via mailbox, if any
                        05DD  1630          ;
                        05DD  1631          ;
         50    DD       05DD  1632          PUSHL   R0                      ; Save table entry address
                        05DF  1633          ;
58   00'A0    3C       05DF  1634          MOVZWL  B^REASON_W_MBX(R0),R8   ; Get mailbox MSG$_xxx code
         52    B5       05E3  1635          TSTW    R2                      ; Any user data ?
         0F    19       05E5  1636          BLSS    40$                     ; If LSS then don't send a message
                        05E7  1637          UPDATE  L,R2,NDC+NDC$L_BRC(R5)  ; Bump "bytes received"
      FA0A'    30       05F3  1638 30$:     BSBW    NET$SEND_CS_MBX         ; Notify user
      FA07'    30       05F6  1639 40$:     BSBW    NET$PURG_RUN            ; Cleanup if exiting RUN state
                        05F9  1640          ;
         50 8ED0       05F9  1641          POPL    R0                      ; Restore table entry address
51   00'A0    3C       05FC  1642          MOVZWL  B^REASON_W_DR(R0),R1    ; Setup second IOSB longword value
50   00'A0    3C       0600  1643          MOVZWL  B^REASON_W_SS(R0),R0    ; Setup first IOSB longword value
      F9F9'    31       0604  1644          BRW     NET$CMPL_ACC            ; Complete IO$_ACCESS if its still
                        0607  1645          ; pending and return to change state
                        0607  1646
```

```
                     0607  1648          .SBTTL  ACT$RCV_DTACK   - DATA ACK message processing
                     0607  1649          .SBTTL  ACT$RCV_LIACK   - INT/LI ACK message processing
                     0607  1650          .SBTTL  NET$PIG_ACK     - Common piggy-backed ACK processing
                     0607  1651 ;*
                     0607  1652 ;
                     0607  1653 ;  If the ACK value is within range, the subchannel block is updated. For valid
                     0607  1654 ;  NAK's, the value of the 'last segment xmitted' is always updated since the
                     0607  1655 ;  remote node is requesting retransmissions.  If poaaible, LNX can be advanced
                     0607  1656 ;  on ACKs to prevent retransmitting unnecessarily.
                     0607  1657 ;
                     0607  1658 ;  The ACK is completely processed.  A user IRP for DATA or INT message may be
                     0607  1659 ;  completed as a result.  'Piggy-backed' ACKs are considered to be independent
                     0607  1660 ;  from the remainder of the message.  Any errors encountered with respect to
                     0607  1661 ;  the ACK number being out of range are not reported to the calling routine.
                     0607  1662 ;
                     0607  1663 ;  Since 'piggy-backed' ACKS are common, The following code is optimized
                     0607  1664 ;  to exit with minimal processing in the expected  case that the ACK value has
                     0607  1665 ;  already been seen.  Optimization also considers NAKs to be rare events.
                     0607  1666 ;
                     0607  1667 ;  INPUTS:  R8    If NET$PIG_ACK   - Ptr to LSB
                     0607  1668 ;                 If ACT$RCV...ACK - Garbage
                     0607  1669 ;           R7    Scratch
                     0607  1670 ;           R6    Msg CXB address
                     0607  1671 ;           R5    XWB address
                     0607  1672 ;           R4    Scratch
                     0607  1673 ;           R3    ACK field value
                     0607  1674 ;           R2    Number of bytes in message not yet accounted for
                     0607  1675 ;           R1    If ACT$RCV_ACK then ptr to ACK field in message
                     0607  1676 ;                 Else pointer to SEG field following the ACK
                     0607  1677 ;           R0    Scratch
                     0607  1678 ;           SP    Caller's return address
                     0607  1679 ;         4(SP)   Caller's caller's return address
                     0607  1680 ;
                     0607  1681 ;  OUTPUTS: R8    LSB address
                     0607  1682 ;           R7    Garbage
                     0607  1683 ;           R4    Garbage
                     0607  1684 ;           R3    If ACT$RCV_ACK then garbage
                     0607  1685 ;                 Else the value of SEG field following the ACK
                     0607  1686 ;           R2    Decremented by 2 since piggy-backed ACK field is
                     0607  1687 ;                 optional and therefore had not yet been accounted for
                     0607  1688 ;           R1    Advance by two bytes
                     0607  1689 ;           R0    Garbage
                     0607  1690 ;
                     0607  1691 ;           All other registers are preserved.
                     0607  1692 ;
                     0607  1693 ;--
                     0607  1694                  .ENABL  LSB
                     0607  1695 NET$PIG_ACK:                                 ; Piggy-backed ACK processing
     52   02    C2   0607  1696          SUBL    #2,R2                      ; Account for ACK field
          08    19   060A  1697          BLSS    10$                        ; If LSS then msg is too small
          31    10   060C  1698          BSBB    PROC_ACK                   ; Process the ACK
     53   81    B0   060E  1699          MOVW    (R1)+,R3                   ; Get SEGNUM field
          F4    19   0611  1700          BLSS    NET$PIG_ACK                ; If LSS, it's another ACK
                05   0613  1701          RSB                                ; Done
                     0614  1702
          8E    D5   0614  1703 10$:     TSTL    (SP)+                      ; Pop caller's address
     57   00'8F 9A   0616  1704          MOVZBL  #NETEVT$_PROERR,R7         ; Setup new event
```

```
                F9E3'  31   061A  1705          BRW     NET$PRE_EMPT            ; Pre-empt with new event
                       061D  1706                                                ;
                       061D  1707 ACT$RCV_LIACK::                                ; INT/LI ACK message processing
          58   00D4 C5   9E  061D  1708          MOVAB   XWB$T_LI(R5),R8         ; Get LSB
               05   11   0622  1709          BRB     15$                     ; Continue
                       0624  1710                                                ;
                       0624  1711 ACT$RCV_DTACK::                               ; DATA ACK message processing
          58   00A4 C5   9E  0624  1712          MOVAB   XWB$T_DT(R5),R8         ; Get LSB
               53   81   B0  0629  1713 15$:     MOVW    (R1)+,R3                ; Get ACK field
                    11   10  062C  1714          BSBB    PROC_ACK                ; Process it
               52   02   C2  062E  1715          SUBL    #2,R2                   ; Is there a 2nd ACK field
                    F6   13  0631  1716          BEQL    15$                     ; If EQL yes
                    05   05  0633  1717          RSB                             ; Else, done
                       0634  1718                                                ;
            58   2C A8   D0  0634  1719 XCHAN:   MOVL    LSB$L_CROSS(R8),R8      ; Get cross-channel LSB
                 09   10  0638  1720          BSBB    20$                     ; Process the ACK
            58   2C A8   D0  063A  1721          MOVL    LSB$L_CROSS(R8),R8      ; Get original LSB
                 05   05  063E  1722          RSB                             ;
                       063F  1723                                                ;
                       063F  1724 PROC_ACK:                                      ; Process ACK value
          F1 53    0D   E0  063F  1725          BBS     #NSP$V_ACK_XCH,R3,XCHAN ; If BS, cross channel ACK
57   53    F000 8F   AB  0643  1726 20$:     BICW3   #^X<F000>,R3,R7         ; Get ACK'd segment number
  54   57    06 A8   A3  0649  1727          SUBW3   LSB$W_HAR(R8),R7,R4     ; Get distance to high ACK rcv'd
               05   12  064E  1728          BNEQ    50$                     ; If EQL, we've seen it before
          01 53    0C   E0  0650  1729          BBS     #NSP$V_ACK_NAK,R3,50$   ; If BS its a NAK, process it
               05   05  0654  1730 40$:     RSB                             ; Done
                       0655  1731                                                ;
                       0655  1732                                                ;
54   54    0C   00   EE  0655  1733 50$:     EXTV    #0,#12,R4,R4            ; MUST SIGN EXTEND
               F8   19  065A  1734          BLSS    40$                     ; If LSS we saw this before
 50    08 A8   57   A3  065C  1735          SUBW3   R7,LSB$W_HAA(R8),R0     ; Greater than 'highest ACK acceptable'
       EF 50    0B   E0  0661  1736          BBS     #11,R0,40$              ; If BS yes,  we can't take it (it must
                       0665  1737                                                ; be old or a race in "segment" flow
                       0665  1738                                                ; control which will resolve itself).
                       0665  1739                                                ; ...branch to enforce LSB Rule 7a.
                       0665  1740          ;
                       0665  1741          ;
                       0665  1742          ;    The ACK value is 'legal', i.e, it can be processed.
                       0665  1743          ;
                       0665  1744          ;
       06 A8    57   B0  0665  1745          MOVW    R7,LSB$W_HAR(R8)        ; Advance HAR
 50    04 A8    57   A3  0669  1746          SUBW3   R7,LSB$W_HXS(R8),R0     ; LEQ than 'highest Xmt-able seg' ?
       04 50    0B   E1  066E  1747          BBC     #11,R0,60$              ; If BC yes
       04 A8    57   B0  0672  1748          MOVW    R7,LSB$W_HXS(R8)        ; Else, advance HXS
                       0676  1749                                                ; ...enforces LSB Rule 1a.
 50    02 A8    57   A3  0676  1750 60$:     SUBW3   R7,LSB$W_LNX(R8),R0     ; Greater than 'last number xmt'd' ?
       04 50    0B   E0  067B  1751          BBS     #11,R0,70$              ; If BS yes, advance LNX
                       067F  1752                                                ; ...enforces LSB Rule 2a.
       04 53    0C   E1  067F  1753          BBC     #NSP$V_ACK_NAK,R3,80$   ; Always update LNX if legal NAK
       02 A8    57   B0  0683  1754 70$:     MOVW    R7,LSB$W_LNX(R8)        ; Reset "Last Number Xmtted"
                       0687  1755 80$:     ASSUME  LSB$V_LI_EQ_0           ;
          28 2B A8   E8  0687  1756          BLBS    LSB$B_STS(R8),PROC_LIACK; If LBS, LS/INT subchanel
                       068B  1757                                                ;
                       068B  1758                  .DSABL  LSB
```

```
                                068B  1760          .SBTTL  PROC_DTACK      - Process of DATA ACK
                                068B  1761 ;++
                                068B  1762 ;
                                068B  1763 ;  The DATA subchannel block is updated according to the ACK value received.
                                068B  1764 ;  Newly ACK'd segements are (conditionally) deallocated and as many user
                                068B  1765 ;  transmit IRPs as possible are completed.
                                068B  1766 ;
                                068B  1767 ;
                                068B  1768 ;  INPUTS:  R8 = DATA subchannel (LSB) pointer
                                068B  1769 ;           R7 = New HAR value
                                068B  1770 ;           R5 = XWB pointer
                                068B  1771 ;           R4 = Number of 'new' ACKs received
                                068B  1772 ;           R3 = ACK field from message
                                068B  1773 ;           R0 = Scratch
                                068B  1774 ;
                                068B  1775 ;  OUTPUTS: R7 = Garbage
                                068B  1776 ;           R4 = Garbage
                                068B  1777 ;           R3 = Garbage
                                068B  1778 ;           R0 = Garbage
                                068B  1779 ;
                                068B  1780 ;           All other registers preserved
                                068B  1781 ;
                                068B  1782 ;--
                                068B  1783 PROC_DTACK:                                  ; DATA subchannel ACK processing
                          7E 51 7D  068B  1784          MOVQ    R1,-(SP)              ; Save regs
                                068E  1785                                            ;
                             54 D5  068E  1786          TSTL    R4                    ; Any new segment's ACK'd ?
                             17 13  0690  1787          BEQL    100$                  ; If EQL no, must be a NAK
                                0692  1788          ;
                                0692  1789          ;
                                0692  1790          ;    See if timed segment has been ACK'd
                                0692  1791          ;
                                0692  1792          ;
                                0692  1793          ASSUME  XWB$V_STS_TID  EQ 0
                                0692  1794
                          11 0E A5 E9 0692  1795          BLBC    XWB$W_STS(R5),40$     ; If LBC timer is unowned
                       0C 0E A5 01 E0 0696  1796          BBS     #XWB$V_STS_TLI,XWB$W_STS(R5),40$ ; If BS, owned by LI channel
                    50 57 48 A5 A3 069B  1797          SUBW3   XWB$W_TIM_ID(R5),R7,R0 ; Prepare 12 bit compare
                       03 50 0B E0 06A0  1798          BBS     #11,R0,40$            ; If BS, owner seg # is larger
                             0D19 30 06A4  1799          BSBW    TIMED_SEG_ACKED       ; Timed segment has been ACK'd
                                06A7  1800 40$:        ;
                                06A7  1801             ;
                                06A7  1802             ;    ACK the CXB's and determine new "HXS" value
                                06A7  1803             ;
                                06A7  1804             ;
                                   3E 10 06A7  1805          BSBB    NET$ACK_XMT_SEGS      ; Cleanup IRP's, CXB's, etc.
                             52    58 D0 06A9  1806 100$:    MOVL    R8,R2                 ; Setup LSB address
                                0228 30 06AC  1807          BSBW    CALC_HXS_LUX          ; Calculate new HXS value
                                06AF  1808                                            ;
                             51    8E 7D 06AF  1809          MOVQ    (SP)+,R1              ; Restore regs
                                   05 06B2  1810          RSB                           ; Done
                                06B3  1811
```

```
                     06B3  1813          .SBTTL  PROC_LIACK      - Process INT/LS ACK
                     06B3  1814 ;++
                     06B3  1815 ;
                     06B3  1816 ;  The INT/LS subchannel state is updated according to the ACK value.  If an
                     06B3  1817 ;  Interrupt message which has been ACK'd then it is posted for completion.
                     06B3  1818 ;
                     06B3  1819 ;
                     06B3  1820 ;  INPUTS:  R8 = INT/LS subchannel (LSB) pointer
                     06B3  1821 ;           R7 = New HAR value
                     06B3  1822 ;           R5 = XWB pointer
                     06B3  1823 ;           R4 = Number of 'new' ACKs received
                     06B3  1824 ;           R3 = ACK field from message
                     06B3  1825 ;           R0 = Scratch
                     06B3  1826 ;
                     06B3  1827 ;  OUTPUTS: R7 = Garbage
                     06B3  1828 ;           R4 = Garbage
                     06B3  1829 ;           R3 = Garbage
                     06B3  1830 ;           R0 = Garbage
                     06B3  1831 ;
                     06B3  1832 ;           All other preserved
                     06B3  1833 ;
                     06B3  1834 ;--
                     06B3  1835 PROC_LIACK:                                          ; Process INT/LS ACKs
          7E   51 7D 06B3  1836          MOVQ    R1,-(SP)                         ; Save regs
                     06B6  1837                                                   ;
          01   54 D1 06B6  1838          CMPL    R4,#1                            ; Is the next msg being ACK'd
               28 12 06B9  1839          BNEQ    100$                             ; If LSSU then no
      6C A5    10 AA 06BB  1840          BICW    #NSP$M_FLW_INUSE,XWB$B_X_FLW(R5) ; Free the LI message slot
      1C A5    10 AA 06BF  1841          BICW    #XWB$M_FLG_SLI,XWB$W_FLG(R5)     ; Nothing left to send for now
                     06C3  1842          ;
                     06C3  1843          ;
                     06C3  1844          ;    See if timed segment has been ACK'd
                     06C3  1845          ;
                     06C3  1846          ;
                     06C3  1847          ASSUME  XWB$V_STS_TID  EQ 0
                     06C3  1848
        08 0E A5   E9 06C3  1849          BLBC    XWB$W_STS(R5),20$                ; If LBC timer is unowned
   03 0E A5    01   E1 06C7  1850          BBC     #XWB$V_STS_TLI,XWB$W_STS(R5),20$ ; If BC, owned by DATA channel
           0CF1   30 06CC  1851          BSBW    TIMED_SEG_ACKED                  ; Cleanup and handoff timer
   0C 6C A5    05   E1 06CF  1852 20$:    BBC     #NSP$V_FLW_INT,XWB$B_X_FLW(R5),90$ ; If BC, not "Interrupt" msg
                     06D4  1853          ;
                     06D4  1854          ;
                     06D4  1855          ;    "ACK" the Interrupt segment and complete the user Xmt IRP if
                     06D4  1856          ;    possible.  If there is another interrupt message and the flow
                     06D4  1857          ;    control allows, then schedule the message for transmission
                     06D4  1858          ;
                     06D4  1859          ;
          0A A8    97 06D4  1860          DECB    LSB$B_X_REQ(R8)                  ; Remote request completed
      57  10 A8    D0 06D7  1861          MOVL    LSB$L_X_PND(R8),R7               ; Get the associated IRP
          50   01   7D 06DB  1862          MOVQ    #SS$_NORMAL,R0                   ; I/O completion status w/o size
               4B   10 06DE  1863          BSBB    XMT_REQ_DONE                     ; Complete the user I/O
             0189   30 06E0  1864 90$:     BSBW    CHK_INT_AVL_R8                   ; Try to set XWB$V_FLG_IAVL
                     06E3  1865                                                   ;
          51   8E   7D 06E3  1866 100$:    MOVQ    (SP)+,R1                         ; Restore regs
                     05 06E6  1867          RSB                                      ; Done
                     06E7  1868
```

```
                     06E7  1870          .SBTTL  NET$ACK_XMT_SEGS  - ACK Xmt Segs, Complete User Xmt IRP's
                     06E7  1871 ;++
                     06E7  1872 ;
                     06E7  1873 ;  ACK each CXB and remove if from the list.  If CXB$B_CODE=0 then then
                     06E7  1874 ;  deallocate it.  The next newly ACK'd CXB is always the first CXB in the
                     06E7  1875 ;  list.
                     06E7  1876 ;
                     06E7  1877 ;  If the byte quota and transmit-packet-window constrains allow, complete all
                     06E7  1878 ;  pending user xmit IRP's.
                     06E7  1879 ;
                     06E7  1880 ;
                     06E7  1881 ;  INPUTS:       R8      DATA channel LSB pointer
                     06E7  1882 ;                R5      XWB pointer
                     06E7  1883 ;                R4      Number of new segments ACK'd -- must be GTR 0
                     06E7  1884 ;                R3-R0   Scratch
                     06E7  1885 ;
                     06E7  1886 ;  OUTPUTS:      R4-R0  Garbage
                     06E7  1887 ;
                     06E7  1888 ;                All other registers are preserved
                     06E7  1889 ;
                     06E7  1890 ;-
                     06E7  1891 NET$ACK_XMT_SEGS::                                           ; ACK new segments
1C A5   0400 8F   AA 06E7  1892          BICW    #XWB$M_FLG_WDAT,XWB$W_FLG(R5)              ; Clear flag
     50   18 A8   D0 06ED  1893 10$:     MOVL    LSB$L_X_CXB(R8),R0                         ; Get next CXB
  18 A8   10 A0   D0 06F1  1894          MOVL    CXB$L_LINK(R0),LSB$L_X_CXB(R8)             ; Remove CXB from list
        0D A8   97 06F6  1895          DECB    LSB$B_X_CXBACT(R8)                         ; Account for it
     5A A5   03   B3 06F9  1896          BITW    #XWB$M_PRO_SFC!XWB$M_PRO_NFC,XWB$B_PRO(R5) ; Msg flow control ?
                05   12 06FD  1897          BNEQ    20$                                    ; If NEQ no
  03 4E A0   06   E1 06FF  1898          BBC     #NSP$V_DATA_EOM,CXB$B_X_NSPTYP(R0),30$     ; If BS, end of message
        0A A8   97 0704  1899 20$:     DECB    LSB$B_X_REQ(R8)                            ; One more request done
                     0707  1900 30$:     ;
                     0707  1901          ;
                     0707  1902          ;    Clear the ACK-outstanding flag.  If there are no more flags set
                     0707  1903          ;    then the CXB is idle and we can either queue it to the free queue
                     0707  1904          ;    or, if we are over-quota, deallocate it.  If any flags remain set,
                     0707  1905          ;    then decrement the current CXB count since the CXB will be
                     0707  1906          ;    deallocated when the final flag eventually clears.
                     0707  1907          ;
                     0707  1908          ;
  0B A0    02   8A 0707  1909          BICB    #CXB$M_CD_ACK,CXB$B_CODE(R0)               ; "ACK" the segment
           0A   12 070B  1910          BNEQ    40$                                        ; If NEQ, still on some
                070D  1911                                                             ; datalink's xmt queue
0E A8    0F A8   91 070D  1912          CMPB    LSB$B_X_CXBCNT(R8),LSB$B_X_CXBQUO(R8)      ; Within CXB quota ?
           08   1B 0712  1913          BLEQU   50$                                        ; If GTRU, over quota
         F8E9'  30 0714  1914          BSBW    NET$DEALLOCATE                             ; Deallocate CXB in R0
        0F A8   97 0717  1915 40$:     DECB    LSB$B_X_CXBCNT(R8)                         ; CXB no longer in use
           05   11 071A  1916          BRB     60$                                        ; Continue
 0118 C5   60   0E 071C  1917 50$:     INSQUE  (R0),XWB$Q_FREE_CXB(R5)                    ; Queue CXB
        C9 54   F5 0721  1918 60$:     SOBGTR  R4,10$                                     ; Loop for each new ACK
                0724  1919                                                             ;
   53   14 A8   D0 0724  1920 100$:    MOVL    LSB$L_X_IRP(R8),R3                         ; Get first IRP
           28   12 0728  1921          BNEQ    CHK_XMT_DONE                               ; If NEQ then got one
                05 072A  1922          RSB                                                ; Else, done
                072B  1923
```

```
                        072B  1925
                        072B  1926 XMT_REQ_DONE:
                        072B  1927              ;
                        072B  1928              ;
                        072B  1929              ;   Setup IRP for completion, move it to 'done' queue, and attempt
                        072B  1930              ;   I/O completion.
                        072B  1931              ;
                        072B  1932              ;
    38 A7   50    7D    072B  1933              MOVQ    R0,IRP$L_IOST1(R7)                      ; Set IOSB image
                        072F  1934 XMT_REQ_DONE_OK:
3A A7    32 A7   B0     072F  1935              MOVW    IRP$W_BCNT(R7),IRP$L_IOST1+2(R7)        ; Size of transfer
0E A8    04 A7   82     0734  1936              SUBB    IRP$B_QUO(R7),LSB$B_X_CXBQUO(R8)        ; Reclaim donated quota
   50    14 A8   9E     0739  1937              MOVAB   LSB$L_X_IRP(R8),R0                      ; Perpare for scan
      51    50   D0     073D  1938 30$:         MOVL    R0,R1                                   ; Make a copy
      50    60   D0     0740  1939              MOVL    (R0),R0                                 ; Get next IRP
            F8   12     0743  1940              BNEQ    30$                                     ; If NEQ, not last
   10 A8    67   D0     0745  1941              MOVL    (R7),LSB$L_X_PND(R8)                    ; Remove IRP from 'PND'
            67   D4     0749  1942              CLRL    (R7)                                    ; Zero it's linkage
      61    57   D0     074B  1943              MOVL    R7,(R1)                                 ; Attach it to 'IRP'
   53    14 A8   D0     074E  1944              MOVL    LSB$L_X_IRP(R8),R3                      ; Get first IRP
                        0752  1945
                        0752  1946 CHK_XMT_DONE:                                              ; Check Xmt IRP list
                        0752  1947              ;
                        0752  1948              ;
                        0752  1949              ;   Complete all the  xmit IRP's allowed by the CXB quota and
                        0752  1950              ;   transmit-packet-window and CXB contraints.
                        0752  1951              ;
                        0752  1952              ;   If the number of unACK'd CXB's is GTRU the current packet-window
                        0752  1953              ;   then completing the I/O now may serve to cause the user to queue
                        0752  1954              ;   another transmit request and thus consume pool for CXB's which
                        0752  1955              ;   cannot be sent.  Hence, in order to avoid using pool unnecessarily,
                        0752  1956              ;   don't complete the I/O request until the number of unACK'd CXB's
                        0752  1957              ;   is LEQU the packet-window.
                        0752  1958              ;
                        0752  1959              ;
0F 5A A5    04    E0    0752  1960              BBS     #XWB$V_PRO_NAR,XWB$B_PRO(R5),NET$XMT_DONE ; If BS, keep pipeline
                        0757  1961                                                              ; as full as possible
0C A8    0D A8   91     0757  1962              CMPB    LSB$B_X_CXBACT(R8),LSB$B_X_PKTWND(R8)   ; Within packet-window ?
         07      1A     075C  1963              BGTRU   200$                                    ; If GTRU then no
0E A8    0D A8   91     075E  1964 110$:        CMPB    LSB$B_X_CXBACT(R8),LSB$B_X_CXBQUO(R8)   ; Within CXB quota ?
         01      1B     0763  1965              BLEQU   NET$XMT_DONE                            ; If LEQU yes
                 05     0765  1966 200$:        RSB                                             ; Done
                        0766  1967
                        0766  1968 NET$XMT_DONE::
         55     DD     0766  1969              PUSHL   R5                                      ; Save XWB pointer
                        0768  1970                                                              ;
   14 A8    63    D0     0768  1971 10$:        MOVL    IRP$L_IOQFL(R3),LSB$L_X_IRP(R8)         ; Detach IRP
   55    1C A3   D0     076C  1972              MOVL    IRP$L_UCB(R3),R5                        ; Get UCB address
00000000'GF    16     0770  1973              JSB     G^COM$POST                              ; Post IRP
   53    14 A8   D0     0776  1974              MOVL    LSB$L_X_IRP(R8),R3                      ; Get next IRP
         EC      12     077A  1975              BNEQ    10$                                     ; If EQL then none left
                        077C  1976                                                              ;
         55 8ED0        077C  1977              POPL    R5                                      ; Recover XWB address
                 05     077F  1978              RSB                                             ; Done
                        0780  1979
```

```
                        0780  1981          .SBTTL  ACT$RCV_LI      - Receive INT/LS message
                        0780  1982 ;++
                        0780  1983 ;
                        0780  1984 ;  Process a received Interrupt or Link Service message.  The format of the
                        0780  1985 ;  message is given below.
                        0780  1986 ;
                        0780  1987 ;
                        0780  1988 ;        15              8 7              0
                        0780  1989 ;        +---------------+---------------+
                        0780  1990 ;        | flw ctl value |     flags     |
                        0780  1991 ;        +---------------+---------------+
                        0780  1992 ;
                        0780  1993 ;                        bit 0  set to turn on DATA backpressure
                        0780  1994 ;                        bit 1  set to turn off DATA      "
                        0780  1995 ;                        bit 2  set if 'flw ctl value' for INT/LS
                        0780  1996 ;                               clear if       "              for DATA
                        0780  1997 ;
                        0780  1998 ;  INPUTS:       R8,R7   Scratch
                        0780  1999 ;                R6      CXB address
                        0780  2000 ;                R5      XWB address
                        0780  2001 ;                R4,R3   Scratch
                        0780  2002 ;                R2      Number of as yet unaccounted bytes in message
                        0780  2003 ;                R1      Pointer to first unparsed byte in message
                        0780  2004 ;                R0      Scratch
                        0780  2005 ;
                        0780  2006 ;  OUTPUTS:      R6      CXB address or "0" if CXB is consumed
                        0780  2007 ;                R5      XWB address
                        0780  2008 ;                R0      Standard VMS status code
                        0780  2009 ;
                        0780  2010 ;                R8,R7,R4,R3,R2,R1 are garbage, all others are unmodified
                        0780  2011 ;
                        0780  2012 ;--
                        0780  2013 ACT$RCV_LI::                                 ; Receive LINK SERVICE messages
                        0780  2014                  ;
                        0780  2015                  ;*************************************************************************
                        0780  2016                  ;
                        0780  2017                  ; NOTE: Since this buffer may be owned by the remote end of the logical
                        0780  2018                  ;       link if both ends of the link are on the local node, the CXB
                        0780  2019                  ;       contents, starting with the NSP header, cannot be modified.
                        0780  2020                  ;
                        0780  2021                  ;*************************************************************************
                        0780  2022                  ;
  1C A5   04   A8       0780  2023          BISW    #XWB$M_FLG_SIACK,XWB$W_FLG(R5)  ; ALWAYS send an ACK
58  00D4 C5   9E        0784  2024          MOVAB   XWB$T_LI(R5),R8                 ; Get INT/LS LSB
                        0789  2025                  ;
                        0789  2026                  ;
                        0789  2027                  ;    Process optional ACK and required SEGMENT NUMBER fields
                        0789  2028                  ;
                        0789  2029                  ;
    53   81   B0        0789  2030          MOVW    (R1)+,R3                        ; Get SEG field
         03   18        078C  2031          BGEQ    10$                             ; If LSS then really ACK field
      FE76    30        078E  2032          BSBW    NET$PIG_ACK                     ; Process the ACK field
                        0791  2033                                                  ; - returns to caller's caller
                        0791  2034                                                  ;   on detected errors
    56   DD             0791  2035 10$:     PUSHL   R6                              ; Save reg - CXB is never
                        0793  2036                                                  ; consumed here
                        0793  2037                                                  ;
```

```
                50    53    26 A8    A3   0793  2038          SUBW3   LSB$W_HAX(R8),R3,R0                     ; Distance from high ACK xmt'd
          50    50    0C    00    EE   0798  2039          EXTV    #0,#12,R0,R0                            ; MUST SIGN EXTEND
                            50    B7   079D  2040          DECW    R0                                      ; Is this the next seq number ?
                            50    19   079F  2041          BLSS    40$                                     ; If LSS, we've seen this before
                            43    14   07A1  2042          BGTR    30$                                     ; If GTR, seq # is too advanced
                4C 39 A6    05    E1   07A3  2043          BBC     #NSP$V_MSG_INT,CXB$B_R_NSPTYP(R6),50$ ; If BC, LINK SERVICE msg
                                       07A8  2044          ;
                                       07A8  2045          ;
                                       07A8  2046          ;    Received message is an INTERRUPT message.  Validate INTERRUPT data
                                       07A8  2047          ;    and move it to the user's mailbox.
                                       07A8  2048          ;
                                       07A8  2049          ;
                      10    52    D1   07A8  2050          CMPL    R2,#16                                  ; Check size of interrupt data
                            39    1A   07AB  2051          BGTRU   30$                                     ; If GTR then illegal
                         29 A8    95   07AD  2052          TSTB    LSB$B_R_CXBQUO(R8)                      ; Can we accept this ?
                            3F    13   07B0  2053          BEQL    40$                                     ; If EQL no, link is running
                                       07B2  2054                                                          ; down (don't NAK if Phase II)
                                       07B2  2055          UPDATE  L,R2,NDC+NDC$L_BRC(R5)                  ; Bump "bytes received"
                                       07BE  2056          ;
                      7E    71    90   07BE  2057          MOVB    -(R1),-(SP)                             ; Backup ptr, save its contents
                      61    52    90   07C1  2058          MOVB    R2,(R1)                                 ; Setup count field
                            52    96   07C4  2059          INCB    R2                                      ; R2 should be total length
                      58    35    D0   07C6  2060          MOVL    #MSG$_INTMSG,R8                         ; Setup mbx msg type
                         F834'  30   07C9  2061          BSBW    NET$SEND_CS_MBX                         ; Build and send mbx msg
                      61    8E    90   07CC  2062          MOVB    (SP)+,(R1)                              ; Restore clobbered cell
                                       07CF  2063          ;
                0275  8F    50    B1   07CF  2064          CMPW    R0,#SS$_NOMBX!1                         ; If 'success' implicit due to
                            0D    13   07D4  2065          BEQL    20$                                     ;  no mbx, ACK INT message but
                                       07D6  2066                                                          ;  don't flow control another
                   0D 50    E9   07D6  2067          BLBC    R0,30$                                  ; If LBC, assume mailbox is full
          1C A5   2000 8F    A8   07D9  2068          BISW    #XWB$M_FLG_SIFL,XWB$W_FLG(R5)           ; Flow control another INT msg
                  00FD C5    97   07DF  2069          DECB    XWB$T_LI+CSB$B_R_CXBQUO(R5)             ; And use our quota for this one
                     006D    31   07E3  2070  20$:     BRW     120$                                    ; ACK the INT message
                                       07E6  2071  30$:     ;
                                       07E6  2072          ;
                                       07E6  2073          ;    Cause a NAK to be sent if partner is phase II.
                                       07E6  2074          ;
                                       07E6  2075          ;
                06 5A A5    02    E1   07E6  2076          BBC     #XWB$V_PRO_PH2,XWB$B_PRO(R5),40$ ; If BC, not Phase II
             0E A5    0200 8F    A8   07EB  2077          BISW    #XWB$M_STS_LINAK,XWB$W_STS(R5)          ; Schedule the NAK message
                      0071    31   07F1  2078  40$:     BRW     140$                                    ; Continue
                                       07F4  2079  50$:     ;
                                       07F4  2080          ;
                                       07F4  2081          ;    Process received LINK SERVICE message
                                       07F4  2082          ;
                                       07F4  2083          ;
          54    81    F0 8F    8B   07F4  2084          BICB3   #NSP$M_FLW_DRV,(R1)+,R4                 ; Mask out driver internal bits
                50    81    98   07F9  2085          CVTBL   (R1)+,R0                                ; Get flow value
                   0B 54    02    E1   07FC  2086          BBC     #NSP$V_FLW_LISUB,R4,60$                 ; If BC, for DATA subchannel
                                       0800  2087          ;
             52    00D4 C5    9E   0800  2088          MOVAB   XWB$T_LI(R5),R2                         ; Use INT/LS subchannel
                53    6F'AF    9E   0805  2089          MOVAB   B^CHK_INT_AVL,R3                        ; Setup action routine for LI
                      14    11   0809  2090          BRB     70$                                     ; Continue
                                       080B  2091          ;
             52    00A4 C5    9E   080B  2092  60$:     MOVAB   XWB$T_DT(R5),R2                         ; Get subchannel block
             53    08A9'CF    9E   0810  2093          MOVAB   W^NEW_DATA_FLOW,R3                      ; Setup action routine address
          13 5A A5    00    E0   0815  2094          BBS     #XWB$V_PRO_NFC,XWB$B_PRO(R5),90$ ; If BS, "no flow" control
```

```
                                                     H 5
04 5A A5   01   E0  081A  2095          BBS    #XWB$V_PRO_SFC,XWB$B_PRO(R5),80$ ; If BS, "segment flow" control
                    081F  2096                                                  ; Else,  "message flow" control
                    081F  2097                                                  ;
          50   95   081F  2098 70$:     TSTB   R0                               ; Check flow control value
          C3   19   0821  2099          BLSS   30$                              ; Negative values are illegal
   50  0A A2   80   0823  2100 80$:     ADDB   LSB$B_X_REQ(R2),R0               ; Okay to add to current count ?
          BD   1D   0827  2101          BVS    30$                              ; If overflow -- ignore msg
   0A A2  50   90   0829  2102          MOVB   R0,LSB$B_X_REQ(R2)               ; Else, setup new X_REQ
                    082D  2103 90$:     ;
                    082D  2104          ;
                    082D  2105          ;    Call action routine with:
                    082D  2106          ;
                    082D  2107          ;       R5 = XWB
                    082D  2108          ;       R2 = LSB
                    082D  2109          ;
                    082D  2110          ;    On return, all but R0,R1,R2,R3 must be preserved.
                    082D  2111          ;
                    082D  2112          ;
                    082D  2113          ASSUME  NSP$V_FLW_XOFF  EQ  0           ; Make sure 'XOFF' is low bit
                    082D  2114          ;
      10 54   E9    082D  2115          BLBC   R4,100$                          ; If LBC the 'XOFF' bit is clear
1C A5   0040 8F   A8  0830  2116          BISW   #XWB$M_FLG_WBP,XWB$W_FLG(R5)     ; Backpressure our transmitter
   02 54   02   E1  0836  2117          BBC    #NSP$V_FLW_LISUB,R4,95$          ; If BC, we're on DATA sub-chan
           63   16  083A  2118          JSB    (R3)                             ; Call LI action routine now
   53  86'AF   9E  083C  2119 95$:     MOVAB  B^SHRINK_XPW,R3                  ; Setup new action routine
   0D 54   01   E1  0840  2120 100$:    BBC    #NSP$V_FLW_XON,R4,110$           ; If BC, 'XON' bit is clear
08 1C A5   06   E5  0844  2121          BBCC   #XWB$V_FLG_WBP,XWB$W_FLG(R5),110$ ; Relax backpressure
03 0E A5   00   E0  0849  2122          BBS    #XWB$V_STS_TID,XWB$W_STS(R5),110$ ; If BS timer is in use
         0BA0   30  084E  2123          BSBW   CANCEL_TIMER                     ; Start timer on any msg that
                    0851  2124                                                  ;  needs it -- clobbers R0
           63   16  0851  2125 110$:    JSB    (R3)                             ; Update request count
                    0853  2126 120$:    ;
                    0853  2127          ;
                    0853  2128          ;    Update LSB$W_HAX on the LI channel
                    0853  2129          ;
                    0853  2130          ;
00FA C5    B6   0853  2131          INCW    XWB$T_LI+LSB$W_HAX(R5)              ; Update ACK value to send
F000 8F    AA   0857  2132          BICW   #^X<F000>,-                          ; Mask off junk bits
00FA C5         085B  2133                  XWB$T_LI+LSB$W_HAX(R5)              ;
00FA C5    B0   085E  2134          MOVW    XWB$T_LI+LSB$W_HAX(R5),-            ; Also the highest msg rcv'd
00F8 C5         0862  2135                  XWB$T_LI+LSB$W_HNR(R5)              ;
                0865  2136                                                      ;
50    01   90   0865  2137 140$:    MOVB   #1,R0                               ; Set success
                0868  2138                                                      ;
      56 8ED0   0868  2139          POPL    R6                                  ; Restore reg
           05   086B  2140          RSB                                         ; Done
                086C  2141
```

```
                     086C  2143          .SBTTL  CHK_INT_AVL     - Conditionally set XWB$V_FLG_IAVL
                     086C  2144          .SBTTL  CHK_INT_AVL_R8  - Conditionally set XWB$V_FLG_IAVL
                     086C  2145 ;+
                     086C  2146 ;
                     086C  2147 ;  The routine is called after an Interrupt flow control message is received.
                     086C  2148 ;  If the new flow control count is non-zero, and if there are Interrupt
                     086C  2149 ;  messages queued for transmission but with no sequence number yet assigned,
                     086C  2150 ;  then and interrupt message is scheduled for transmission.
                     086C  2151 ;
                     086C  2152 ;
                     086C  2153 ;  INPUTS:  R5  XWB address
                     086C  2154 ;           R2  INT/LS subchannel LSB address
                     086C  2155 ;           R0  Scratch
                     086C  2156 ;
                     086C  2157 ;  OUTPUTS: R2  Garbage
                     086C  2158 ;
                     086C  2159 ;           All registers are preserved.
                     086C  2160 ;
                     086C  2161 ;
                     086C  2162 ;-
                     086C  2163 CHK_INT_AVL_R8:                                  ; Try to set XWB$M_FLG_IAVL
          52   58    D0  086C  2164          MOVL    R8,R2                       ; Setup LSB pointer
                     086F  2165 CHK_INT_AVL:                                     ; Try to set XWB$M_FLG_IAVL
            0A A2    95  086F  2166          TSTB    LSB$B_X_REQ(R2)             ; Any Interrupt msg requested ?
              11     13  0872  2167          BEQL    10$                         ; If EQL then no
       50   10 A2    D0  0874  2168          MOVL    LSB$L_X_PND(R2),R0          ; Get pending IRP
              0B     13  0878  2169          BEQL    10$                         ; If EQL then none
    06 20 A0    06    E1  087A  2170          BBC     #IO$V_INTERRUPT,IRP$W_FUNC(R0),10$ ; If BC, already used
1C A5    1000 8F    A8  087F  2171          BISW    #XWB$M_FLG_IAVL,XWB$W_FLG(R5)   ; Flag need to build INT msg
                     05  0885  2172 10$:     RSB                                 ; Done
                         0886  2173
```

```
                                                J 5
0886  2175          .SBTTL  SHRINK_XPW      - Shrink the DATA transmit-packet-window
0886  2176          .SBTTL  NEW_DATA_FLOW   - React to flow control msg
0886  2177 ;+
0886  2178 ;
0886  2179 ;  The DATA channels Link Subchannel Block (LSB) is processed in conjunction
0886  2180 ;  with the transmitter's flow control type and the transmitter's IRP queue to
0886  2181 ;  determine the value of the highest transmittable segment.  This value
0886  2182 ;  replaces LSB$W_HXS.
0886  2183 ;
0886  2184 ;
0886  2185 ;  INPUTS:  R5  XWB address
0886  2186 ;           R3  Scratch
0886  2187 ;           R2  Scratch
0886  2188 ;           R1  Scratch
0886  2189 ;           R0  Scratch
0886  2190 ;
0886  2191 ;  OUTPUTS: R5  Preserved
0886  2192 ;           R3  Garbage
0886  2193 ;           R2  DATA channel LSB address
0886  2194 ;           R1  Garbage
0886  2195 ;           R0  Garbage
0886  2196 ;
0886  2197 ;           All other registers are preserved.
0886  2198 ;
0886  2199 ;
0886  2200 ;-
0886  2201
0886  2202 ;
0886  2203 ;   The following two routines are the only routines which may result in
0886  2204 ;   lowering the value of LSB$W_HXS.
0886  2205 ;
0886  2206
0886  2207 SHRINK_XPW:                                          ; Shrink DATA xmt-packet-window
   52   00A4 C5   9E  0886  2208          MOVAB   XWB$T_DT(R5),R2                 ; Setup R2
     0C A2    02   86  088B  2209          DIVB    #2,LSB$B_X_PKTWND(R2)           ; Cut it in half
  0B 5A A5    04   E1  088F  2210          BBC     #XWB$V_PRO_NAR,XWB$B_PRO(R5),20$; If BC, almost done
7E   0C A2    02   87  0894  2211          DIVB3   #2,LSB$B_X_PKTWND(R2),-(SP)      ; Get 1/4 of original
     0C A2    8E   80  0899  2212          ADDB    (SP)+,LSB$B_X_PKTWND(R2)        ; And add it in
              03   12  089D  2213          BNEQ    30$                             ; If NEQ, okay
        0C A2      96  089F  2214 20$:      INCB    LSB$B_X_PKTWND(R2)              ; Add 1 to prevent zero
 F759 CF    03   85  08A2  2215 30$:      MULB3   #3,NSP$B_ADJ_XPW,-              ; Get adjustment threshold
        0B A2      08A7  2216                  LSB$B_X_ADJ(R2)                 ; Reset threshold (trebled
                   08A9  2217                                                  ; since we were unsuccessful)
                   08A9  2218                                                  ; Fall thru
                   08A9  2219 NEW_DATA_FLOW:                                   ; Respond to new DATA_FLOW msg
              2C   10  08A9  2220          BSBB    CALC_HXS_LUX                    ; Calculate new HXS value
                   08AB  2221          ;
                   08AB  2222          ;
                   08AB  2223          ;   Since we may be reducing the LSB$W_HXS value, we cannot assume the
                   08AB  2224          ;   validity of LSB Rule 4a. until the next few instructions have been
                   08AB  2225          ;   executed.
                   08AB  2226          ;
                   08AB  2227          ;   Enforce LSB Rule 4a. here by reducing LSB$W_LNX if LSB$W_HXS was
                   08AB  2228          ;   reduced below the current LSB$W_LNX value.
                   08AB  2229          ;
                   08AB  2230          ;
50   51   02 A2   A3  08AB  2231          SUBW3   LSB$W_LNX(R2),R1,R0             ; Prepare 12 bit compare
```

```
                                                K 5
                08 50    0B   E1  08B0  2232          BBC    #11,R0,70$                          ; If BC, LNX leq HXS
                                  08B4  2233                                                     ; Else, HXS lss LNX  (illegal)
                02 A2    51   B0  08B4  2234          MOVW    R1,LSB$W_LNX(R2)                   ; ...enforces LSB rule 4a.
                1C A5    20   A8  08B8  2235          BISW   #XWB$M_FLG_WHGL,XWB$W_FLG(R5)       ; Set wait flag
                                  08BC  2236 70$:      ;
                                  08BC  2237           ;
                                  08BC  2238           ;    If the timer ticking on the DATA subchannel then it needs to be
                                  08BC  2239           ;    cancelled if HXS has just shrunk below the segment currently
                                  08BC  2240           ;    being timed, or if the link is now backpressured off.
                                  08BC  2241           ;
                                  08BC  2242           ;
                                  08BC  2243           ASSUME  XWB$V_STS_TID  EQ  0                ;
                                  08BC  2244                                                      ;
                  16 0E A5   E9   08BC  2245           BLBC    XWB$W_STS(R5),100$                 ; If LBC, no msg being timed
               11 0E A5    01   E0  08C0  2246           BBS    #XWB$V_STS_TLI,XWB$W_STS(R5),100$  ; If BS, LI channel has timer
            09 1C A5    06   E0  08C5  2247           BBS    #XWB$V_FLG_WBP,XWB$W_FLG(R5),90$   ; If BS, backpressured
         50    51    48 A5   A3  08CA  2248           SUBW3   XWB$W_TIM_ID(R5),R1,R0             ; Prepare for 12 bit compare
               03 50    0B   E1  08CF  2249           BBC    #11,R0,100$                         ; If BC, timed i.d. is leq HXS
                     0B1B    30  08D3  2250 90$:      BSBW    CANCEL_TIMER                       ; Timer is available for any
                                  08D6  2251                                                     ; channel which  can use it
                             05   08D6  2252 100$:     RSB                                        ; Return
                                  08D7  2253
```

```
                        08D7  2255          .SBTTL  CALC_HXS...     - Calc 'highest xmt seg sendable'
                        08D7  2256 ;+
                        08D7  2257 ;
                        08D7  2258 ;  The DATA channels Link Subchannel Block (LSB) is processed in conjunction
                        08D7  2259 ;  with the transmitter's flow control type and the transmitter's IRP queue to
                        08D7  2260 ;  determine the value of the highest transmittable segment (HXS)
                        08D7  2261 ;
                        08D7  2262 ;
                        08D7  2263 ;  INPUTS:  R5  XWB address
                        08D7  2264 ;           R3  Scratch
                        08D7  2265 ;           R2  DATA channel LSB address
                        08D7  2266 ;           R1  Current LSB$W_LUX value
                        08D7  2267 ;           R0  Scratch
                        08D7  2268 ;
                        08D7  2269 ;  OUTPUTS: R5  Preserved
                        08D7  2270 ;           R3  Garbage
                        08D7  2271 ;           R2  Preserved
                        08D7  2272 ;           R1  New LSB$W_HXS value
                        08D7  2273 ;           R0  Garbage
                        08D7  2274 ;
                        08D7  2275 ;           All other registers are preserved.
                        08D7  2276 ;
                        08D7  2277 ;
                        08D7  2278 ;-
                        08D7  2279 CALC_HXS_LUX:                                ; Calc HXS, process FLG_WHGL
           03DE   30    08D7  2280          BSBW    FILL_XMT_CXBS               ; Try to fill some
                        08DA  2281 CALC_HXS_XMT:                                ;
     51    62   B0      08DA  2282          MOVW    LSB$W_LUX(R2),R1            ; Get LUX value
           00   E0      08DD  2283          BBS    #XWB$V_PRO_NFC,-             ; If BS, "no flow" control
   48 5A A5              08DF  2284                  XWB$B_PRO(R5),30$           ;   (most commonly used)
     53  0A A2   9A      08E2  2285          MOVZBL  LSB$B_X_REQ(R2),R3          ; Get number of seg/msg requests
           01   E0      08E6  2286          BBS    #XWB$V_PRO_SFC,-             ; If BS, "segment flow" control
     21 5A A5              08E8  2287                  XWB$B_PRO(R5),20$           ; Else,  "message flow" control
                        08EB  2288          ;
                        08EB  2289          ;
                        08EB  2290          ;   Message flow control
                        08EB  2291          ;
                        08EB  2292          ;   R3 contains number of 'end-of-message' segments requested but not
                        08EB  2293          ;   yet ACKed.  Find the number of the highest segment queued within
                        08EB  2294          ;   this limit.
                        08EB  2295          ;
                        08EB  2296          ;
     51    06 A2   B0      08EB  2297          MOVW    LSB$W_HAR(R2),R1            ; Preset R1 assuming X_REQ was zero
                        08EF  2298                                              ; -- this gets us to 30$ with the
                        08EF  2299                                              ;    correct value in R1.
     50    08 A2   9E      08EF  2300          MOVAB   -CXB$L_LINK -           ;
                        08F3  2301                  +LSB$L_X_CXB(R2),R0         ; Prepare for CXB scan
           12   11      08F3  2302          BRB     10$                         ; Go to end of loop
     50    10 A0   D0      08F5  2303 5$:      MOVL    CXB$L_LINK(R0),R0           ; Get next segment
           2F   13      08F9  2304          BEQL    30$                         ; If EQL then none left
        F000 8F   AB      08FB  2305          BICW3  #^X<F000>,-              ;
     51    55 A0           08FF  2306                  CXB$W_X_NSPSEQ(R0),R1       ; Setup 'highest seg sendable'
           06   E1      0902  2307          BBC    #NSP$V_DATA_EOM,-            ;
       EE 4E A0            0904  2308                  CXB$B_X_NSPTYP(R0),5$       ; If BC, not end of message
          EB 53   F4      0907  2309 10$:     SOBGEQ  R3,5$                       ; Loop for each message requested
           1E   11      090A  2310          BRB     30$                         ; Continue
                        090C  2311 20$:     ;
```

```
                                090C  2312          ;
                                090C  2313          ;    Segment flow control
                                090C  2314          ;
                                090C  2315          ;    R3 contains the number of segments requested but not yet ACKed.
                                090C  2316          ;    Determine thenumber of the highest segment queued with this limit.
                                090C  2317          ;
                                090C  2318          ;
          53    06 A2    A0     090C  2319          ADDW    LSB$W_HAR(R2),R3        ; Calc. highest seg requested
        50  53    51    A3     0910  2320          SUBW3   R1,R3,R0                ; Prepare 12 bit compare
          12 50    0B    E1     0914  2321          BBC     #11,R0,30$              ; If BC, LUX leq the high seg requested
    51    53    F000 8F    AB   0918  2322          BICW3   #^X<F000>,R3,R1         ; Else use high seg requested
          53    08 A2    A2     091E  2323          SUBW    LSB$W_HAA(R2),R3        ; Have we already sent a larger seg?
          04 53    0B    E1     0922  2324          BBC     #11,R3,30$              ; If BC no
          08 A2    51    B0     0926  2325          MOVW    R1,LSB$W_HAA(R2)        ; Else yes, we must have just received
                                092A  2326                                          ; negative flow control credits.  Reset
                                092A  2327                                          ; HAA to make appear that we've never
                                092A  2328                                          ; sent the excess segments.
                                092A  2329 30$:     ;
                                092A  2330          ;
                                092A  2331          ;    Calculate the number of the highest segment we're allowed to send
                                092A  2332          ;    based on the transmitter-packet-window.  Use the minimum of this
                                092A  2333          ;    value and the value allowed by flow control.
                                092A  2334          ;
                                092A  2335          ;    In order to increase pipelining to the maximum extent allowed by
                                092A  2336          ;    the network capacity (which is always changing), everytime the
                                092A  2337          ;    packet-window was a factor in reducing HXS we decrement a counter.
                                092A  2338          ;    When that counter reaches zero we open up packet-window by 1.  If
                                092A  2339          ;    it is opened too far then some other event (e.g., a need to
                                092A  2340          ;    retransmit) will cause it to close again.
                                092A  2341          ;
                                092A  2342          ;       NOTE:   The adjustment counter is decremented whenever HXS is
                                092A  2343          ;               greater than or EQUAL to HAR plus the currenty window
                                092A  2344          ;               value.  The EQUAL is important since HXS is also limited
                                092A  2345          ;               (see above) by the number of available segments and the
                                092A  2346          ;               number of available segments is limited by the rate of
                                092A  2347          ;               I/O completion to the user which is limited by the
                                092A  2348          ;               packet window value (refer to routine CHK_XMT_DONE).
                                092A  2349          ;
                                092A  2350          ;               Hence, the window value and the number of available
                                092A  2351          ;               segments would limit each other unless pressure is
                                092A  2352          ;               applied to open the window when the size of the window
                                092A  2353          ;               is in equilibrium with the amount of data available.
                                092A  2354          ;
                                092A  2355          ;
                                092A  2356          ASSUME  NSP$C_MAX_XPW  LE  254  ; Make sure it can fit in a byte
                                092A  2357                                          ;
          53    0C A2    9A     092A  2358          MOVZBL  LSB$B_X_PKTWND(R2),R3   ; Get transmit-packet-window value
          53    06 A2    A0     092E  2359          ADDW    LSB$W_HAR(R2),R3        ; Add in last ACK value received
    53    F000 8F    AA         0932  2360          BICW    #^X<F000>,R3            ; Mask off junk bits
    50    51    53    A3         0937  2361          SUBW3   R3,R1,R0                ; Prepare for 12 bit compare
       19 50    0B    E0         093B  2362          BBS     #11,R0,40$              ; If BS, tentative HXS lss HAR+window
                                093F  2363          ;
                                093F  2364          ;
                                093F  2365          ;    The packet window has restricted the amount of data which we can
                                093F  2366          ;    send.  Therefore, try to increase the packet window.
                                093F  2367          ;
                                093F  2368          ;
```

```
                51   53    B0   093F   2369          MOVW    R3,R1                           ; Update HXS value
                F6BB CF    91   0942   2370          CMPB    NSP$B_MAX_XPW,-                 ; Are we already at the maximum ?
                  0C A2           0946   2371                  LSB$B_X_PRTWND(R2)              ;
                  0E   1B   0948   2372          BLEQU   40$                             ; If LEQU then yes
                  0B A2    97   094A   2373          DECB    LSB$B_X_ADJ(R2)                 ; Another need to adjust window detected
                  09   12   094D   2374          BNEQ    40$                             ; If NEQ, don't adjust it yet
                F6AD CF    90   094F   2375          MOVB    NSP$B_ADJ_XPW,-                 ; Reset threshold
                  0B A2           0953   2376                  LSB$B_X_ADJ(R2)                 ;
                  0C A2    96   0955   2377          INCB    LSB$B_X_PKTWND(R2)              ; Open the window by one packet
                               0958   2378 40$:       ;
                               0958   2379            ;
                               0958   2380            ;    Setup the new HXS value and the WHGL flag
                               0958   2381            ;
                               0958   2382            ;
            04 A2    51   B0   0958   2383          MOVW    R1,LSB$W_HXS(R2)                ; Setup new HXS
            02 A2    51   B1   095C   2384          CMPW    R1,LSB$W_LNX(R2)                ; HXS < LNX ?  (LNX never > HXS)
                     05   13   0960   2385          BEQL    200$                            ; If EQL no
            1C A5    20   AA   0962   2386          BICW    #XWB$M_FLG_WHGL,XWB$W_FLG(R5)   ; Clear wait condition
                     05   0966   2387          RSB                                     ; Done
                          0967   2388
            1C A5    20   A8   0967   2389 200$:    BISW    #XWB$M_FLG_WHGL,XWB$W_FLG(R5)   ; Set "wait for HXS gtr LNX"
                     05   096B   2390          RSB                                     ; Done
                          096C   2391
```

```
                        096C  2393
                        096C  2394                      .ENABL  LSB
                        096C  2395
                        096C  2396 NEW_RCV_IRP:
    OBED'CF    9F   096C  2397              PUSHAB  W^RCV_COPY                      ; Setup intercept routine addr.
 50   1C A8    D0   0970  2398              MOVL    LSB$L_R_IRP(R8),R0              ; Get first IRP
           24    12   0974  2399              BNEQ    200$                            ; If NEQ none
 1C A8     53    D0   0976  2400              MOVL    R3,LSB$L_R_IRP(R8)              ; This IRP is the first one
 56   20 A8    D0   097A  2401 60$:         MOVL    LSB$L_R_CXB(R8),R6              ; Get first CXB
           19    13   097E  2402              BEQL    100$                            ; If EQL, none
20 A8    10 A6    D0   0980  2403              MOVL    CXB$L_LINK(R6),LSB$L_R_CXB(R8)  ; Remove CXB from list
        28 A8    97   0985  2404              DECB    LSB$B_R_CXBCNT(R8)              ; Account for it
 52    0C A6    3C   0988  2405              MOVZWL  CXB$W_LENGTH(R6),R2             ; Get amount of data
 54    39 A6    9A   098C  2406              MOVZBL  CXB$B_R_NSPTYP(R6),R4           ; Get message flags
          00FC   30   0990  2407              BSBW    CXB_TO_IRP                      ; Attach CXB to this IRP
                        0993  2408 RCV_IRP:
 53   1C A8    D0   0993  2409              MOVL    LSB$L_R_IRP(R8),R3              ; Get IRP
           E1    12   0997  2410              BNEQ    60$                             ; If NEQ, got one
                 05   0999  2411 100$:        RSB                                     ; Done
                        099A  2412
                        099A  2413
    51    50    D0   099A  2414 200$:        MOVL    R0,R1                           ; Copy IRP address
    50    61    D0   099D  2415              MOVL    (R1),R0                         ; Get next IRP
           F8    12   09A0  2416              BNEQ    200$                            ; If NEQ, got one
    61    53    D0   09A2  2417              MOVL    R3,(R1)                         ; Else, attach this IRP
                 05   09A5  2418              RSB                                     ; Done
                        09A6  2419                      .DSABL  LSB
                        09A6  2420
```

```
                                09A6  2422          .SBTTL  ACT$RCV_DATA    - Process rcv'd DATA message
                                09A6  2423 ;++
                                09A6  2424 ;
                                09A6  2425 ;  A received data segment is processed.  If it is acceptable then the
                                09A6  2426 ;  IRP's message buffer (CXB) is moved to the LSB.
                                09A6  2427 ;
                                09A6  2428 ;
                                09A6  2429 ;  INPUTS:       R8,R7   Scratch
                                09A6  2430 ;                R6      CXB address
                                09A6  2431 ;                R5      XWB address
                                09A6  2432 ;                R4,R3   Scratch
                                09A6  2433 ;                R2      Number of as yet unaccounted bytes in message
                                09A6  2434 ;                R1      Pointer to first unparsed byte in message
                                09A6  2435 ;                R0      Scratch
                                09A6  2436 ;
                                09A6  2437 ;  OUTPUTS:      R6      CXB address or  "0"  if CXB is consumed
                                09A6  2438 ;                R5      XWB address
                                09A6  2439 ;                R0      Standard VMS status code
                                09A6  2440 ;
                                09A6  2441 ;                R8,R7,R4,R3,R2,R1 are garbage, all others are unmodified
                                09A6  2442 ;
                                09A6  2443 ;--
                                09A6  2444                          .ENABL  LSB
                                09A6  2445 OVF:                                             ; Process overflow segment
                                09A6  2446                  ;
                                09A6  2447                  ;
                                09A6  2448                  ;   This is the next logical segment but we cannot take it since the
                                09A6  2449                  ;   receiver is in the "overflow" state.   While we are in this state,
                                09A6  2450                  ;   continue ACKing and discarding message segments up to and including
                                09A6  2451                  ;   the next "end-of-message" segment.  As soon as the next "end-of-
                                09A6  2452                  ;   message" is processed, exit from the "overflow" state.
                                09A6  2453                  ;
                                09A6  2454                  ;
          26 A8    53    B0     09A6  2455          MOVW    R3,LSB$W_HAX(R8)                ; Becomes "highest ack xmitted"
          24 A8    53    B0     09AA  2456          MOVW    R3,LSB$W_HNR(R8)                ; Becomes "highest number rcvd"
          06 50    06    E1     09AE  2457          BBC     #NSP$V_DATA_EOM,R0,10$          ; If BC then not last message
    0E A5    0080  8F    AA     09B2  2458          BICW    #XWB$M_STS_OVF,XWB$W_STS(R5)    ; Clear overflow status
                   14    11     09B8  2459 10$:     BRB     30$                             ; Cause ACK to be sent
                                09BA  2460
                                09BA  2461 NOT_NEXT:                                        ; Segment arrived out of order
                                09BA  2462                  ;
                                09BA  2463                  ;
                                09BA  2464                  ;   This segment was not the next one expected.   If the segment number
                                09BA  2465                  ;   was larger than expected, it is probably due to congestion loss in
                                09BA  2466                  ;   the Routing Layer -- for that reason, don't send a NAK since we
                                09BA  2467                  ;   may contribute to congestion at a time when the network needs to
                                09BA  2468                  ;   slow down its traffic rate.
                                09BA  2469                  ;
                                09BA  2470                  ;   If the segment number was less than expected, then it is probably
                                09BA  2471                  ;   a retransmission.   Send an ACK since the original one could have
                                09BA  2472                  ;   gotten lost due to congestion.
                                09BA  2473                  ;
                                09BA  2474                  ;
    54   54    0C    00    EE   09BA  2475          EXTV    #0,#12,R4,R4                    ; Is it already buffered ?
                   00    15     09BF  2476          BLEQ    NO_BUF                          ; If LEQ then yes
                                09C1  2477                  ;   Put caching here
                                09C1  2478 NO_BUF:                                          ; Cannot take buffer
```

```
                                09C1  2479          ;
                                09C1  2480          ;
                                09C1  2481          ;   We cannot take the buffer, either because the datalink layer needs
                                09C1  2482          ;   it or because the user process is over its allowed outstanding
                                09C1  2483          ;   segment count.
                                09C1  2484          ;
                                09C1  2485          ;   If the partner is phase II we must send a NAK.
                                09C1  2486          ;
                                09C1  2487          ;
                          10    09C1  2488          BSBB    BACK_PRESSURE                           ; Request XOFF to be sent
                    06 5A A5    E1  09C3  2489          BBC     #XWB$V_PRO_PH2,XWB$B_PRO(R5),30$ ; If BC, not Phase II
                  0E A5   0100  8F  A8  09C8  2490          BISW    #XWB$M_STS_DTNAK,XWB$W_STS(R5)  ; Send a NAK on next ACK
                       1C A5    08  A8  09CE  2491  30$:    BISW    #XWB$M_FLG_SDACK,XWB$W_FLG(R5)  ; Cause ACK to be sent
                                05  09D2  2492  40$:    RSB                                     ; Done
                                09D3  2493
                                09D3  2494
                                09D3  2495  BACK_PRESSURE:                                       ; Back-pressure remote xmitter
                                09D3  2496          ;
                                09D3  2497          ;
                                09D3  2498          ;   If the XWB$L_PID field is zero, then there is no current owner for
                                09D3  2499          ;   this link.  It is in the RUN state trying to transmit the data
                                09D3  2500          ;   message currently committed to the pipeline.  However, the fact
                                09D3  2501          ;   that we've gotten a receive that we cannot buffer means that the
                                09D3  2502          ;   link is about to deadlock since the absence of an owner process
                                09D3  2503          ;   implies that we'll never be able to buffer it.  Therefore, if the
                                09D3  2504          ;   XWB$L_PID field is zero then mark the link for disconnect.
                                09D3  2505          ;
                                09D3  2506          ;   NOTE:  This works in conjuction with the backround timer no-op
                                09D3  2507          ;          flow control messages in routine T_O_RUN.
                                09D3  2508          ;
                                09D3  2509          ;
                    34 A5       D5  09D3  2510          TSTL    XWB$L_PID(R5)                   ; Any owner process ?
                       03       12  09D6  2511          BNEQ    50$                             ; If NEQ then yes
                     09D9       30  09D8  2512          BSBW    NET$MARK_LINK                   ; Else cause link to disconnect
                                09DB  2513  50$:    ;
                                09DB  2514          ;
                                09DB  2515          ;   If the remote transmitter is already back-pressured, then cancel
                                09DB  2516          ;   any attempt to toggle its state.  Else, toggle its state.
                                09DB  2517          ;
                                09DB  2518          ;   Back-pressure the remote transmitter ONLY if there are currently
                                09DB  2519          ;   no receive IRP's linked to the LSB (otherwise there would be no
                                09DB  2520          ;   way to know when to relax the back-pressure).
                                09DB  2521          ;
                                09DB  2522          ;
                    1C A8       D5  09DB  2523          TSTL    LSB$L_R_IRP(R8)                 ; Any IRP's ?
                       11       12  09DE  2524          BNEQ    60$                             ; If NEQ yes, don't send XOFF
                1C A5   0800 8F  AA  09E0  2525          BICW    #XWB$M_FLG_TBPR,XWB$W_FLG(R5)   ; Assume no message needed
                  06 0E A5    06  E0  09E6  2526          BBS     #XWB$V_STS_RBP, XWB$W_STS(R5),60$; If BS, already back-pressured
                1C A5   0800 8F  A8  09EB  2527          BISW    #XWB$M_FLG_TBPR,XWB$W_FLG(R5)   ; Send back-pressure message
                                05  09F1  2528  60$:    RSB                                     ; Done
                                09F2  2529
                                09F2  2530          .DSABL  LSB
```

```
                                09F2  2532
                                09F2  2533 ACT$RCV_DATA::                                  ; Process rcv'd DATA msg
          58    00A4 C5    9E   09F2  2534          MOVAB   XWB$T_DT(R5),R8                ; Get DATA LSB
                                09F7  2535          ;
                                09F7  2536          ;
                                09F7  2537          ;    Process optional ACK and required SEGMENT NUMBER fields
                                09F7  2538          ;
                                09F7  2539          ;
                53    81    B0  09F7  2540          MOVW    (R1)+,R3                       ; Get SEG field
                      03    18  09FA  2541          BGEQ    10$                            ; If LSS, really ACK field
                    FC08    30  09FC  2542          BSBW    NET$PIG_ACK                    ; Parse and process the ACK
             05 53    0E    E1  09FF  2543 10$:     BBC     #NSP$V_SEQ_NAR,R3,15$          ; If BC, no ACK suppression
          39 A6    80    8F  88  0A03  2544          BISB    #NSP$M_DATA_NAR,CXB$B_R_NSPTYP(R6) ; Else, allow ACK suppression
             53    F000  8F  AA  0A08  2545 15$:     BICW    #^X<F000>,R3                   ; Mask off segment # junk bits
       54    53      24  A8  A3  0A0D  2546          SUBW3   LSB$W_HNR(R8),R3,R4            ; Distance from 'high seg rcvd'
    01 54    0C      00  EC  0A12  2547          CMPV    #0,#12,R4,#1                   ; Is this next in sequence ?
                          A1  12  0A17  2548          BNEQ    NOT_NEXT                       ; If NEQ then no
          88 0E A5    07  E0  0A19  2549          BBS     #XWB$V_STS_OVF,XWB$W_STS(R5),OVF ; If BS, in overflow state
                                0A1E  2550          ;
                                0A1E  2551          ;
                                0A1E  2552          ;    See how many CXB's NSP currently has buffered.  This does not count
                                0A1E  2553          ;    the ones which have been ACK'd since the session layer owns those.
                                0A1E  2554          ;
                                0A1E  2555          ;    LSB$B_R_CXBQUO goes to zero when we're running the link down.
                                0A1E  2556          ;
                                0A1E  2557          ;       NOTE:   It is not necessary to check if there is an IRP waiting
                                0A1E  2558          ;               for the incoming CXB since the difference between
                                0A1E  2559          ;               LSB$W_HAX and LSB$W_HNR is always zero if there is an
                                0A1E  2560          ;               IRP waiting.
                                0A1E  2561          ;
                                0A1E  2562          ;
                                0A1E  2563          ;
29 A8    28 A8    91  0A1E  2564          CMPB    LSB$B_R_CXBCNT(R8),LSB$B_R_CXBQUO(R8)  ; Already at limit ?
                9C    1E  0A23  2565          BGEQU   NO_BUF                                 ; If GEQU yes
                                0A25  2566          ;
                                0A25  2567          ;
                                0A25  2568          ;    It is important to signal the transmitter to slow down (shrink its
                                0A25  2569          ;    transmit-packet-window if it has one) if it starts consuming too
                                0A25  2570          ;    many unACK'd buffers.
                                0A25  2571          ;
                                0A25  2572          ;    The intent is to help the packet flow rate to stabilize with a
                                0A25  2573          ;    minimum number of unACK'd buffers consumed at both the transmitter
                                0A25  2574          ;    and receiver ends.  The stable packet rate is not affected -- only
                                0A25  2575          ;    the number of buffers required to sustain that rate.
                                0A25  2576          ;
                                0A25  2577          ;    The value chosen as the threshold for signalling the transmitter
                                0A25  2578          ;    must be large enough to accomodate a slight variance in the
                                0A25  2579          ;    session layer receiver rate, but small enough to avoid excess
                                0A25  2580          ;    buffer consumption.  It is NOT the number of buffers consumed when
                                0A25  2581          ;    the stable rate is achieved, but attempts to be an upper limit on
                                0A25  2582          ;    that value.
                                0A25  2583          ;
                                0A25  2584          ;
08 5A A5    04  E0  0A25  2585          BBS     #XWB$V_PRO_NAR,XWB$B_PRO(R5),20$; If BS, allow queue to build
                                0A2A  2586                                             ; to CXBQUO limit.
   28 A8    02  91  0A2A  2587          CMPB    #2,LSB$B_R_CXBCNT(R8)                  ; Is this the 3rd buffer ?
                02  12  0A2E  2588          BNEQ    20$                                ; If NEQ no, continue
```

```
                A1   10   0A30  2589              BSBB    BACK_PRESSURE                       ; Back-pressure and continue
          54    39 A6   90   0A32  2590 20$:      MOVB    CXB$B_R_NSPTYP(R6),R4               ; Get message type
            11  38 A6   E9   0A36  2591            BLBC    CXB$B_R_FLG(R6),40$                 ; If LBC, okay to take the CXB
                                0A3A  2592            ;
                                0A3A  2593            ;
                                0A3A  2594            ;   Clone a new CXB since we cannot take this one.  Complete remaining
                                0A3A  2595            ;   processing via a subroutine, and return to our caller with the
                                0A3A  2596            ;   original CXB pointer intact.
                                0A3A  2597            ;
                                0A3A  2598            ;
              03EC   30   0A3A  2599            BSBW    CLONE_RCV_CXB                       ; Clone a new one
             03  50   E8   0A3D  2600            BLBS    R0,30$                              ; If LBS, okay
              FF7E   31   0A40  2601            BRW     NO_BUF                              ; Else, allocation failure
                 57   DD   0A43  2602 30$:      PUSHL   R7                                  ; Save original CXB address
                                0A45  2603            ;                                   ;
                 04   10   0A45  2604            BSBB    40$                                 ; Complete processing CXB
                                0A47  2605            ;                                   ;
                 56 8ED0   0A47  2606            POPL    R6                                  ; Recover original CXB address
                      05   0A4A  2607            RSB                                         ; Done
                                0A4B  2608
                                0A4B  2609 40$:       ;
                                0A4B  2610            ;
                                0A4B  2611            ;   Complete setting up CXB
                                0A4B  2612            ;
                                0A4B  2613            ;
              10 A6   D4   0A4B  2614            CLRL    CXB$L_LINK(R6)                      ; Init the linked list pointer
          39 A6   54   90   0A4E  2615            MOVB    R4,CXB$B_R_NSPTYP(R6)               ; Save msg type in the CXB
          3A A6   53   B0   0A52  2616            MOVW    R3,CXB$W_R_NSPSEQ(R6)               ; Remember the seg #
          0C A6   52   B0   0A56  2617            MOVW    R2,CXB$W_LENGTH(R6)                 ; Enter length of data
    0E A6     51   56   A3   0A5A  2618            SUBW3   R6,R1,CXB$W_OFFSET(R6)              ; Enter offset to data
                66   51   D0   0A5F  2619            MOVL    R1,(R6)                             ; Enter pointer to data
                                0A62  2620            ;
                                0A62  2621            ;
                                0A62  2622            ;   Consume buffer, update the receiver state.  Attach the CXB to
                                0A62  2623            ;   either an IRP if there is one, or to the LSB if there isn't.
                                0A62  2624            ;
                                0A62  2625            ;   Must check for a none-empty LSB CXB list first to avoid a race
                                0A62  2626            ;   condition which could result in copying segments out of order.
                                0A62  2627            ;
                                0A62  2628            ;
    24 A8    53   B0   0A62  2629            MOVW    R3,LSB$W_HNR(R8)                    ; Becomes new "highest # rcv'd"
    51   20 A8   D0   0A66  2630            MOVL    LSB$L_R_CXB(R8),R1                  ; Any CXB's on LSB?
              0A   12   0A6A  2631            BNEQ    90$                                 ; If so, put this on at the end
    53   1C A8   D0   0A6C  2632            MOVL    LSB$L_R_IRP(R8),R3                  ; Get IRP
              1D   12   0A70  2633            BNEQ    CXB_TO_IRP                          ; If NEQ, got one
                                0A72  2634            ;
                                0A72  2635            ;
                                0A72  2636            ;   Attach CXB to the end of the LSB list
                                0A72  2637            ;
                                0A72  2638            ;
    51   10 A8   9E   0A72  2639            MOVAB   -CXB$L_LINK+LSB$L_R_CXB(R8),R1      ; Prepare for CXB scan
       50    51   D0   0A76  2640 90$:      MOVL    R1,R0                               ; Travel CXB list
    51   10 A0   D0   0A79  2641            MOVL    CXB$L_LINK(R0),R1                   ; Get next CXB
              F7   12   0A7D  2642            BNEQ    90$                                 ; If NEQ then not end of list
              10 A6   D4   0A7F  2643            CLRL    CXB$L_LINK(R6)                      ; Init link-list pointer
    10 A0     56   D0   0A82  2644            MOVL    R6,CXB$L_LINK(R0)                   ; Chain CXB to list
              56   D4   0A86  2645            CLRL    R6                                  ; Consume CXB
```

```
                28 A8   96  0A88  2646          INCB    LSB$B_R_CXBCNT(R8)              ; Account for new CXB
                        05  0A8B  2647          RSB                                     ; Done
                            0A8C  2648
                            0A8C  2649          .DSABL  LSB
                            0A8C  2650
```

```
                                                H  6
                     0A8C  2652
                     0A8C  2653 R7_CXB_TO_IRP:                                  ; Attach R7 CXB to IRP
          56   57   D0  0A8C  2654          MOVL    R7,R6                       ; Pickup original CXB
                     0A8F  2655 CXB_TO_IRP:                                     ; Attach CXB to IRP
                     0A8F  2656          ;
                     0A8F  2657          ;                   Inputs             Outputs
                     0A8F  2658          ;                   ------             -------
                     0A8F  2659          ;           R8      LSB ptr            unchanged
                     0A8F  2660          ;           R7      scratch            garbage
                     0A8F  2661          ;           R6      CXB pointer        0 if consumed, else unchanged
                     0A8F  2662          ;           R5      XWB pointer        unchanged
                     0A8F  2663          ;           R4      Message flags      garbage
                     0A8F  2664          ;           R3      IRP pointer        garbage
                     0A8F  2665          ;           R2      Segment byte count garbage
                     0A8F  2666          ;           R1      scratch            garbage
                     0A8F  2667          ;           R0      scratch            garbage
                     0A8F  2668          ;
                     0A8F  2669          ;
                     0A8F  2670          ;
        3A A3   52   B1  0A8F  2671          CMPW    R2,IRP$L_IOST1+2(R3)       ; Enough buffer space left ?
                66   1B  0A93  2672          BLEQU   50$                        ; If LEQU okay, attach to IRP
     49 20 A3   08   E1  0A95  2673          BBC     #IO$V_MULTIPLE,IRP$W_FUNC(R3),40$ ; If BC, data over-run
                     0A9A  2674          ;
                     0A9A  2675          ;
                     0A9A  2676          ;    The message overflows the user buffer, but the user has requested
                     0A9A  2677          ;    that a partial message be returned without error if needed -- the
                     0A9A  2678          ;    user will issue another read to continue reading the same message.
                     0A9A  2679          ;
                     0A9A  2680          ;    Clone the CXB and adjust each CXB such that one contains the first
                     0A9A  2681          ;    portion of the segment and the other contains the final portion.
                     0A9A  2682          ;    Attach the first to the IRP, and the second to the LSB.
                     0A9A  2683          ;
                     0A9A  2684          ;
  38 A3    0601 8F   B0  0A9A  2685          MOVW    #SS$_BUFFEROVF,IRP$L_IOST1(R3) ; Indicate "partial message"
             0383  30  0AA0  2686          BSBW    CLONE_RCV_CXB_1            ; Clone a new CXB
    3A A7     26 A8   B0  0AA3  2687          MOVW    LSB$W_HAX(R8),CXB$W_R_NSPSEQ(R7); Backup sequence number
                     0AA8  2688          ;                                      ;  in 'first half' CXB
       52     3A A3   3C  0AA8  2689          MOVZWL  IRP$L_IOST1+2(R3),R2       ; Get amount 'first half' data
       0C A7     52   B0  0AAC  2690          MOVW    R2,CXB$W_LENGTH(R7)        ; Setup 'first half' length
       54   0040 8F   A8  0AB0  2691          BISW    #NSP$M_DATA_EOM,R4         ; Trigger IRP I/O completion
              16   50  E9  0AB5  2692          BLBC    R0,20$                     ; If LBC, allocation failure
            66     52  C0  0AB8  2693          ADDL    R2,(R6)                    ; Update 'last half' data ptr
          0E A6    52  A0  0ABB  2694          ADDW    R2,CXB$W_OFFSET(R6)        ; Setup  'last half' offset
          0C A6    52  A2  0ABF  2695          SUBW    R2,CXB$W_LENGTH(R6)        ; Setup  'last half' length
    10 A6    20 A8   D0  0AC3  2696          MOVL    LSB$L_R_CXB(R8),CXB$L_LINK(R6) ; Move CXB to BEGINING of list
       20 A8    56   D0  0AC8  2697          MOVL    R6,LSB$L_R_CXB(R8)         ;
                BE   11  0ACC  2698          BRB     R7_CXB_TO_IRP              ; Loop to process original CXB
                     0ACE  2699 20$:         ;
                     0ACE  2700          ;
                     0ACE  2701          ;    Failed to clone a 'last half' CXB.
                     0ACE  2702          ;
                     0ACE  2703          ;    Routine CXB_TO_IRP is called in one of two cases:  either a new
                     0ACE  2704          ;    CXB has arrived, or a new IRP has arrived.  In the former, the
                     0ACE  2705          ;    LSB$L_R_CXB list should be empty.  In the latter, the LSB$L_R_CXB
                     0ACE  2706          ;    list may be non-empty, and in this case we must drain it since it
                     0ACE  2707          ;    now has a missing CXB (the 'last half' CXB that we couldn't clone).
                     0ACE  2708          ;    Hence, draining the CXB list is either a no-op or it is required.
```

```
                                0ACE  2709          ;
                                0ACE  2710          ;
                                0ACE  2711          ;
                                0ACE  2712          ;    NOTE: On RCV IRP draining, always drain IRP$L_SVAPTE and move
                                0ACE  2713          ;          IRP$L_IOST2 to IRP$L_SVAPTE if its negative and CHAINED is
                                0ACE  2714          ;          clear.
                                0ACE  2715          ;
  24 A8    26 A8    B0          0ACE  2716          MOVW    LSB$W_HAX(R8),LSB$W_HNR(R8)     ; LSB's CXB list is empty
0E A5    0100 8F    A8          0AD3  2717          BISW    #XWB$M_STS_DTNAK,XWB$W_STS(R5)  ; Next Data ACK should be a NAK
  54     0080 8F    AA          0AD9  2718          BICW    #NSP$M_DATA_NAR,R4              ; Trigger sending of ACK
            039E    30          0ADE  2719          BSBW    NET$DRAIN_R_LSBCXB              ; Deallocate all CXB's attached
                                0AE1  2720                                                  ; to the LSB
            A9      11          0AE1  2721          BRB     R7_CXB_TO_IRP                   ; Process original CXB
                                0AE3  2722  40$:     ;
                                0AE3  2723          ;
                                0AE3  2724          ;    Shrink data in CXB down to what we can handle
                                0AE3  2725          ;
                                0AE3  2726          ;
38 A3    0838 8F    B0          0AE3  2727          MOVW    #SS$_DATAOVERUN,IRP$L_IOST1(R3) ; Report data over-run
    06 54    06    E2          0AE9  2728          BBSS    #NSP$V_DATA_EOM,R4,45$          ; If BS, last seg in message
                                0AED  2729                                                  ; (set bit to trigger IRP I/O
                                0AED  2730                                                  ;  completion)
0E A5    0080 8F    A8          0AED  2731          BISW    #XWB$M_STS_OVF,XWB$W_STS(R5)    ; Else, set "overflow flag"
    52    3A A3    3C          0AF3  2732  45$:     MOVZWL  IRP$L_IOST1+2(R3),R2            ; Take as much as we can
    0C A6    52    B0          0AF7  2733          MOVW    R2,CXB$W_LENGTH(R6)             ; Adjust length of data
                                0AFB  2734  50$:     ;
                                0AFB  2735          ;
                                0AFB  2736          ;    Move CXB to IRP
                                0AFB  2737          ;
                                0AFB  2738          ;
    50    2C A3    D0          0AFB  2739          MOVL    IRP$L_SVAPTE(R3),R0             ; Get attached CXB ?
            32     12          0AFF  2740          BNEQ    60$                             ; If NEQ, there was one there
                                0B01  2741          ;
                                0B01  2742          ;
                                0B01  2743          ;    This is the first CXB to be attached to this IRP.
                                0B01  2744          ;
                                0B01  2745          ;
    2C A3    56    D0          0B01  2746          MOVL    R6,IRP$L_SVAPTE(R3)             ; Attach this one
36 2A A3    05    E0          0B05  2747          BBS     #IRP$V_CHAINED,IRP$W_STS(R3),70$ ; If BS, chaining allowed
    32 54    06    E0          0B0A  2748          BBS     #NSP$V_DATA_EOM,R4,70$          ; If BS, this is the first and
                                0B0E  2749                                                  ;  last CXB for this IRP
2D 48 A3    1F    E0          0B0E  2750          BBS     #31,IRP$L_SES_BUF(R3),70$       ; If BS, session buffer present
    51    32 A3    3C          0B13  2751          MOVZWL  IRP$W_BCNT(R3),R1               ; Get size of buffer area needed
    51    5B A1    9E          0B17  2752          MOVAB   NSP$C_HSZ_DATA -                ; Add in overhead
                                0B1B  2753                  +TR3$C_HSZ_DATA -           ;
                                0B1B  2754                  +CXB$C_OVERHEAD(R1),R1      ;
          F4E2'     30          0B1B  2755          BSBW    NET$ALONONPAGED                 ; Allocate buffer
       5D 50      E9          0B1E  2756          BLBC    R0,200$                         ; If LBC, failed
    48 A3    52    D0          0B21  2757          MOVL    R2,IRP$L_SES_BUF(R3)            ; Save buffer address
    62    57 A2    9E          0B25  2758          MOVAB   CXB$T_X_DATA(R2),(R2)           ; Stuff address of the data area
    3C A3    62    D0          0B29  2759          MOVL    (R2),IRP$L_IOST2(R3)            ;  in both the CXB and the IRP
    52    0C A6    3C          0B2D  2760          MOVZWL  CXB$W_LENGTH(R6),R2             ; Recover amount of data in CXB
             0D    11          0B31  2761          BRB     70$                             ; Continue
                                0B33  2762  60$:     ;
                                0B33  2763          ;
                                0B33  2764          ;    Attach new CXB to end of IRP's CXB chain
                                0B33  2765          ;
```

```
                     0B33  2766          ;
          51    50   D0  0B33  2767          MOVL    R0,R1                           ; Copy CXB address
     50   10   A0   D0  0B36  2768          MOVL    CXB$L_LINK(R0),R0               ; Get next CXB
               F7   12  0B3A  2769          BNEQ    60$                             ; If NEQ, contine
     10   A1   56   D0  0B3C  2770          MOVL    R6,CXB$L_LINK(R1)               ; Attach CXB
          10   A6   D4  0B40  2771 70$:     CLRL    CXB$L_LINK(R6)                  ; Terminate linked list
               03   10  0B43  2772          BSBB    150$                            ; Update state variables etc.
               56   D4  0B45  2773          CLRL    R6                              ; Indicate "CXB consumed"
               05   0B47  2774 100$:    RSB                                     ; Done
                    0B48  2775
                    0B48  2776 150$:    ;
                    0B48  2777          ;
                    0B48  2778          ;    CXB has been attached to the IRP.  Update state variables,
                    0B48  2779          ;    byte counts, and schedule an ACK message.
                    0B48  2780          ;
                    0B48  2781          ;
                    0B48  2782          UPDATE  L,R2,NDC+NDC$L_BRC(R5)          ; Add to total for remote node
26   A8     3A   A6   B0  0B54  2783          MOVW    CXB$W_R_NSPSEQ(R6),LSB$W_HAX(R8); Update "highest ack to xmit"
     04   54     07   E0  0B59  2784          BBS     #NSP$V_DATA_NAR,R4,160$         ; If BS, no need to send ACK yet
     1C   A5     08   A8  0B5D  2785          BISW    #XWB$M_FLG_SDACK,XWB$W_FLG(R5)  ; Cause ACK to be sent
               05   A3   96  0B61  2786 160$:    INCB    IRP$B_CXBCNT(R3)                ; Bump CXB count
04   A6     3C   A3   D0  0B64  2787          MOVL    IRP$L_IOST2(R3),4(R6)           ; Save user VA
     3C   A3     52   C0  0B69  2788          ADDL    R2,IRP$L_IOST2(R3)              ; Update user VA pointer
     3A   A3     52   A2  0B6D  2789          SUBW    R2,IRP$L_IOST1+2(R3)            ; Consume bytes
                    0B71  2790          ;
                    0B71  2791          ;
                    0B71  2792          ;    If the IRP CXB count is at its threshold, then dispatch to the
                    0B71  2793          ;    action routine to move the data to the user buffer.  Since moving
                    0B71  2794          ;    the data may involve going to IPL 2, the XWB may not be in the
                    0B71  2795          ;    RUN state upon return.
                    0B71  2796          ;
                    0B71  2797          ;
     14   54     06   E0  0B71  2798          BBS     #NSP$V_DATA_EOM,R4,RCV_DONE     ; If BS, last CXB for this IRP
F488 CF     05   A3   91  0B75  2799          CMPB    IRP$B_CXBCNT(R3),NSP$B_R_CXBTHR ; Too many CXB accumulating?
               70   1E  0B7B  2800          BGEQU   RCV_COPY                        ; If GEQU yes, copy to them
                    0B7D  2801                                                  ; to the user's buffer
               05   0B7D  2802          RSB                                     ; Done
                    0B7E  2803
          2C   A3   D4  0B7E  2804 200$:    CLRL    IRP$L_SVAPTE(R3)                ; Detach CXB
               05   0B81  2805          RSB                                     ; Done
                    0B82  2806
```

```
                                0B82  2808
                                0B82  2809 NET$RCV_DONE::
          03 38 A3      E8      0B82  2810          BLBS    IRP$L_IOST1(R3),RCV_DONE        ; Branch if successful
              02E2      30      0B86  2811          BSBW    NET$DRAIN_R_IRP_CXB             ; Drain all attached CXB's
                                0B89  2812 RCV_DONE:
                                0B89  2813          ;
                                0B89  2814          ;    Detach IRP from LSB.  If there is a Session layer buffer attached
                                0B89  2815          ;    to the IRP (this only happens on non-chained buffer ALTSTART
                                0B89  2816          ;    requests), then the CXB data must first be moved into it.
                                0B89  2817          ;
                                0B89  2818          ;
       15 48 A3       1F      E1 0B89  2819          BBC     #31,IRP$L_SES_BUF(R3),30$       ; If BC, no attached buffer
       37 0C A3       1F      E1 0B8E  2820          BBC     #31,IRP$L_PID(R3),300$          ; If BC, it's a bug
                         58    10 0B93  2821          BSBB    RCV_COPY                        ; Move CXB data to buffer
             53    1C A8       D0 0B95  2822          MOVL    LSB$L_R_IRP(R8),R3              ; Recover IRP
       2C 2C A3       1F      E0 0B99  2823          BBS     #31,IRP$L_SVAPTE(R3),300$       ; If BS bug, CXB's not copied
       2C A3     48 A3        D0 0B9E  2824          MOVL    IRP$L_SES_BUF(R3),IRP$L_SVAPTE(R3) ; Move attached buffer
          1C A8     63        D0 0BA3  2825 30$:     MOVL    (R3),LSB$L_R_IRP(R8)            ; Detach IRP
                       04     13 0BA7  2826          BEQL    50$                             ; If EQL, this is the last IRP
                  FDE6 CF     9F 0BA9  2827          PUSHAB  RCV_IRP                         ; Cause return to intercept
                                0BAD  2828                                                  ; routine
                                0BAD  2829 50$:     ;
                                0BAD  2830          ;
                                0BAD  2831          ;    Send IRP to I/O completion
                                0BAD  2832          ;
                                0BAD  2833          ;
       32 A3     3A A3        A2 0BAD  2834          SUBW    IRP$L_IOST1+2(R3),IRP$W_BCNT(R3); Calc xfer size for IOPOST
       3A A3     32 A3        B0 0BB2  2835          MOVW    IRP$W_BCNT(R3),IRP$L_IOST1+2(R3); Store xfer size for IOSB
                 3C A3        D4 0BB7  2836          CLRL    IRP$L_IOST2(R3)                 ; Zero second IOSB longword
             55  1C A3        D0 0BBA  2837          MOVL    IRP$L_UCB(R3),R5                ; Get UCB address
          00000000'GF         16 0BBE  2838          JSB     G^COM$POST                      ; Another packet for the heap
           55    FF5C C8       9E 0BC4  2839          MOVAB   -XWB$T_DT(R8),R5                ; Recover XWB
                         05     05 0BC9  2840          RSB                                     ; Done
                                0BCA  2841
                                0BCA  2842 300$:    BUG_CHECK  NETNOSTATE,FATAL
                                0BCE  2843
```

```
                        0BCE  2845
                        0BCE  2846                      .ENABL  LSB
                        0BCE  2847
                        0BCE  2848 RCV_COPY2:
          41 5B   E9    0BCE  2849                BLBC    R11,NET$QAST                    ; If LBC, can't go to IPL 2
                        0BD1  2850 RCV_COPY1:
                        0BD1  2851                ;
                        0BD1  2852                ;
                        0BD1  2853                ;   Detaching the CXB is essential since we may go to IPL 2 to probe
                        0BD1  2854                ;   the user buffer and an IPL 8 event may cause all CXB's, IRP's etc
                        0BD1  2855                ;   to be deallocated.
                        0BD1  2856                ;
                        0BD1  2857                ;
  2C A7   10 A6   D0    0BD1  2858                MOVL    CXB$L_LINK(R6),IRP$L_SVAPTE(R7) ; Detach it
          05 A7   97    0BD6  2859                DECB    IRP$B_CXBCNT(R7)                ; Account for it
                        0BD9  2860                ;
                        0BD9  2861                ;
                        0BD9  2862                ;   Get user VA descriptor, copy data from CXB
                        0BD9  2863                ;
                        0BD9  2864                ;
    52    0C A6   3C    0BD9  2865                MOVZWL  CXB$W_LENGTH(R6),R2             ; Get number of bytes
    51    04 A6   D0    0BDD  2866                MOVL    4(R6),R1                        ; Get user address
            01D2  30    0BE1  2867                BSBW    COPY_DATA                       ; Update desc., copy data
          1D 50   E9    0BE4  2868                BLBC    R0,200$                         ; If LBC, error
       50    56   D0    0BE7  2869                MOVL    R6,R0                           ; Prepare for deallocation
            F413' 30    0BEA  2870                BSBW    NET$DEALLOCATE                  ; Deallocate the block
                        0BED  2871 RCV_COPY:
                        0BED  2872                ;
                        0BED  2873                ;
                        0BED  2874                ;   If there is an IRP and its PID field is non-negative, then the
                        0BED  2875                ;   IRP comes from QIO and we must get back to the user process at
                        0BED  2876                ;   IPL 2 to copy the data into the user buffer.
                        0BED  2877                ;
                        0BED  2878                ;   If the PID is negative the IRP came from ALTSTART. If a destination
                        0BED  2879                ;   buffer does not exists, then simply exit without copying the CXBs.
                        0BED  2880                ;
                        0BED  2881                ;
    57    1C A8   D0    0BED  2882                MOVL    LSB$L_R_IRP(R8),R7              ; Get first IRP
            10    13    0BF1  2883                BEQL    100$                            ; If EQL, none
    56    2C A7   D0    0BF3  2884                MOVL    IRP$L_SVAPTE(R7),R6             ; Get next CXB
            0A    13    0BF7  2885                BEQL    100$                            ; If EQL, none
  D0 0C A7    1F  E1    0BF9  2886                BBC     #31,IRP$L_PID(R7),RCV_COPY2     ; If BC, must go to IPL 2
  CE 48 A7    1F  E0    0BFE  2887                BBS     #31,IRP$L_SES_BUF(R7),RCV_COPY1 ; If BS, there's a buffer to
                        0C03  2888                                                        ; copy CXB's into
                  05    0C03  2889 100$:    RSB                                         ; Else, ignore request
                        0C04  2890
                        0C04  2891
    57    1C A8   D1    0C04  2892 200$:    CMPL    LSB$L_R_IRP(R8),R7                  ; Same IRP still there ?
            07    12    0C08  2893                BNEQ    210$                            ; If NEQ, we're done
    38 A7    50   3C    0C0A  2894                MOVZWL  R0,IRP$L_IOST1(R7)              ; Setup status
          32 A7   B4    0C0E  2895                CLRW    IRP$W_BCNT(R7)                  ; Must zero byte-count
                  05    0C11  2896 210$:    RSB                                         ; Done
                        0C12  2897
                        0C12  2898                      .DSABL  LSB
                        0C12  2899
```

```
                        0C12  2901
                        0C12  2902 NET$QAST::                                   ; Schedule Special Kernel AST
          34 A5   D5    0C12  2903          TSTL    XWB$L_PID(R5)               ; XWB still assigned to process?
             14   13    0C15  2904          BEQL    100$                        ; If EQL, no
OE A5   0800 8F   A8    0C17  2905          BISW    #XWB$M_STS_ASTREQ,XWB$W_STS(R5) ; Indicate AST request
  09 0E A5   0A   E2    0C1D  2906          BBSS    #XWB$V_STS_ASTPND,XWB$W_STS(R5),100$ ; Request use of AST block
                        0C22  2907                                              ;
             3F   BB    0C22  2908          PUSHR   #^M<R0,R1,R2,R3,R4,R5>      ; Save regs
          54   01   D0    0C24  2909          MOVL    #1,R4                       ; Use I/O completion priority
               03   10    0C27  2910          BSBB    200$                        ; Fork to get below IPL$_SYNC
               3F   BA    0C29  2911          POPR    #^M<R0,R1,R2,R3,R4,R5>      ; Recover XWB address
                        0C2B  2912                                              ;
               05   0C2B  2913 100$:  RSB                                       ; Done
                        0C2C  2914
   55   0000'C5   9E    0C2C  2915 200$:  MOVAB   XWB$$(R5),R5                  ; Goto AST block context
     0B A5   06   90    0C31  2916          MOVB    #IPL$_QUEUEAST,11(R5)       ; Setup fork level
  00000000'GF   16    0C35  2917          JSB     G^EXE$FORK                  ; Fork
        52   54   D0    0C3B  2918          MOVL    R4,R2                       ; Get priority boost class
 0B A5   80 8F   90    0C3E  2919          MOVB    #^X<80>,ACB$B_RMOD(R5)      ; Setup Special Kernel mode
 18 A5   56'AF   9E    0C43  2920          MOVAB   B^NET$KAST,ACB$L_KAST(R5)   ; Setup AST address
0C A5   0034'C5   D0    0C48  2921          MOVL    -XWB$$+XWB$L_PID(R5),ACB$L_PID(R5) ; Setup PID
               06   13    0C4E  2922          BEQL    NET$KAST                    ; If EQL, link is not accessed
  00000000'GF   17    0C50  2923          JMP     G^SCH$QAST                  ; Queue the Kernel AST
                        0C56  2924
                        0C56  2925 NET$KAST::                                   ; Special Kernel AST service
          0FC0 8F   BB    0C56  2926          PUSHR   #^M<R6,R7,R8,R9,R10,R11>    ; Save regs
                        0C5A  2927          DSBINT  #NET$C_IPL                  ; Go to sychronizing IPL
                        0C60  2928                                              ;
   55   0000'C5   9E    0C60  2929          MOVAB   -XWB$$(R5),R5               ; Got XWB context
   58   00A4 C5   9E    0C65  2930          MOVAB   XWB$T_DT(R5),R8             ; Get LSB
             5B   D4    0C6A  2931          CLRL    R11                         ; Say "can't go to IPL 2"
000C'C5    34 A5   D0    0C6C  2932          MOVL    XWB$L_PID(R5),XWB$$+ACB$L_PID(R5) ; Setup PID, again
               02   13    0C72  2933          BEQL    5$                          ; If EQL, no longer accessed
               5B   D6    0C74  2934          INCL    R11                         ; Say "okay to go to IPL 2"
               68   DD    0C76  2935 5$:     PUSHL   LSB$W_LUX(R8)               ; Save current "last used xmt #"
                        0C78  2936                                              ;
1C A5   0400 8F   AA    0C78  2937          BICW    #XWB$M_FLG_WDAT,XWB$W_FLG(R5) ; Init wait for buffer flag
0E A5   0800 8F   AA    0C7E  2938          BICW    #XWB$M_STS_ASTREQ,XWB$W_STS(R5) ; Clear request flag
  06 1C A5   07   E1    0C84  2939 10$:    BBC     #XWB$V_FLG_SDT,XWB$W_FLG(R5),20$; If BC, not in RUN state
          00AA   30    0C89  2940          BSBW    XMT_COPY                    ; Process transmitter needs
          FF5E   30    0C8C  2941          BSBW    RCV_COPY                    ; Process receiver needs
               0B   E4    0C8F  2942 20$:    BBSC    #XWB$V_STS_ASTREQ,-         ; If BS, new AST request came in
        F0 0E A5            0C91  2943                  XWB$W_STS(R5),10$       ; while we we're at IPL 2
0E A5   0400 8F   AA    0C94  2944          BICW    #XWB$M_STS_ASTPND,XWB$W_STS(R5) ; Release AST block
                        0C9A  2945                                              ;
          50 8ED0   0C9A  2946          POPL    R0                          ; Get former LUX value
  0B 1C A5   07   E1    0C9D  2947          BBC     #XWB$V_FLG_SDT,XWB$W_FLG(R5),30$; If BC, not in RUN state
          68   50   B1    0CA2  2948          CMPW    R0,LSB$W_LUX(R8)            ; Has it changed ?
               06   13    0CA5  2949          BEQL    30$                         ; If EQL, no
          52   58   D0    0CA7  2950          MOVL    R8,R2                       ; Copy LSB address
             FC2D   30    0CAA  2951          BSBW    CALC_HXS_XMT                ; Calc new HXS value
             F350'  30    0CAD  2952 30$:    BSBW    NET$SCH_MSG                 ; Schedule new work
                        0CB0  2953                                              ;
                        0CB0  2954          ENBINT                              ; Restore IPL
          0FC0 8F   BA    0CB3  2955          POPR    #^M<R6,R7,R8,R9,R10,R11>    ; Restore regs
                        0CB7  2956          RSB                                 ; Done
                        0CB8  2957
```

```
                                 0CB8  2959
                                 0CB8  2960 FILL_XMT_CXBS:
          50    10 A2    D0      0CB8  2961          MOVL    LSB$L_X_PND(R2),R0      ; Get pending IRP
                19       13      0CBC  2962          BEQL    30$                     ; If EQL none
                0D A2    91      0CBE  2963          CMPB    LSB$B_X_CXBACT(R2),-    ; Active CXB count within packet window?
                0C A2            0CC1  2964                  LSB$B_X_PKTWND(R2)      ;
                12       1A      0CC3  2965          BGTRU   30$                     ; If GTRU, no
                0D A2    91      0CC5  2966 10$:     CMPB    LSB$B_X_CXBACT(R2),-    ; Active CXB count within quota ?
                0E A2            0CC8  2967                  LSB$B_X_CXBQUO(R2)      ;
                0B       1E      0CCA  2968          BGEQU   30$                     ; If GEQU, can't allocate any more CXB's
                08 5B    E8      0CCC  2969          BLBS    R11,30$                 ; If LBS, okay to go to IPL 2
      04 0C A0     1F    E0      0CCF  2970          BBS     #31,IRP$L_PID(R0),50$   ; If BS then ALTSTART, don't need IPL 2
                                 0CD4  2971                                          ;
              FF3B       30      0CD4  2972          BSBW    NET$QAST                ; Startup AST
                         05      0CD7  2973 30$:     RSB                             ; Done
                                 0CD8  2974
                01DE 8F  BB      0CD8  2975 50$:     PUSHR   #^M<R1,R2,R3,R4,R6,R7,R8>       ; Save regs
                58   52  D0      0CDC  2976          MOVL    R2,R8                           ; Setup LSB pointer
                     55  10      0CDF  2977          BSBB    XMT_COPY                        ; Build xmt segments
                                 0CE1  2978                                                  ;; Any trouble resetting HGL ?
                01DE 8F  BA      0CE1  2979          POPR    #^M<R1,R2,R3,R4,R6,R7,R8>       ; Save regs
                         05      0CE5  2980          RSB                                     ; Done
                                 0CE6  2981
```

```
                        0CE6  2983          .ENABL  LSB
                        0CE6  2984
                        0CE6  2985          ASSUME  IRP$L_IOQFL  EQ  0              ; IRP queue linkage is offset 0
                        0CE6  2986
                        0CE6  2987 300$:    ;
                        0CE6  2988          ;
                        0CE6  2989          ;    Copy failed.  The CXB has been deallocated.
                        0CE6  2990          ;
                        0CE6  2991          ;
          0F A8   97    0CE6  2992          DECB    LSB$B_X_CXBCNT(R8)             ; Account for lost CXB
    10 A8   57    D1    0CE9  2993          CMPL    R7,LSB$L_X_PND(R8)             ; IRP still there ?
          06    12    0CED  2994          BNEQ    100$                           ; If NEQ, no
       32 A7    B4    0CEF  2995          CLRW    IRP$W_BCNT(R7)                 ; Zero eventual transfer size
        FA36    30    0CF2  2996          BSBW    XMT_REQ_DONE                   ; Move to completion queue
                      0CF5  2997                                                 ; with status in R0
          05    0CF5  2998 100$:    RSB                                          ; Done
                      0CF6  2999
        FF19    31    0CF6  3000 400$:    BRW     NET$QAST                       ; Startup kernel mode ast
                      0CF9  3001
1C A5 0400 8F   A8    0CF9  3002 200$:    BISW    #XWB$M_FLG_WDAT,XWB$W_FLG(R5)  ; Flag need to get try again
          05    0CFF  3003          RSB                                          ; Return with LBC in R0
                      0D00  3004
                      0D00  3005 XMT_COPY1:
        E3 50   E9    0D00  3006          BLBC    R0,300$                        ; If LBC, then error
                      0D03  3007          ;
                      0D03  3008          ;
                      0D03  3009          ;    Enter sequence number.
                      0D03  3010          ;
                      0D03  3011          ;
    51  68  01  A1     0D03  3012          ADDW3   #1,LSB$W_LUX(R8),R1            ; Get new 'last used seq #'
    51  F000 8F  AA     0D07  3013          BICW    #^X<F000>,R1                   ; Trim to 12 bits
  55 A6   51    B0     0D0C  3014          MOVW    R1,CXB$W_X_NSPSEQ(R6)          ; Enter it into message
      68    51    B0     0D10  3015          MOVW    R1,LSB$W_LUX(R8)               ; Store new LUX
                      0D13  3016          ;
                      0D13  3017          ;
                      0D13  3018          ;    Attach CXB and request permission to transmit it.
                      0D13  3019          ;
                      0D13  3020          ;
  0B A6   02    90    0D13  3021          MOVB    #CXB$M_CD_ACK,CXB$B_CODE(R6)   ; Say "not yet ACK'ed"
  50    08 A8   9E    0D17  3022          MOVAB   -CXB$L_LINK+LSB$L_X_CXB(R8),R0 ; Prepare for scan
    51    50    D0     0D1B  3023 10$:     MOVL    R0,R1                          ; Save last pointer value
  50    10 A1   D0     0D1E  3024          MOVL    CXB$L_LINK(R1),R0              ; Get next CXB
          F7    12     0D22  3025          BNEQ    10$                            ; If NEQ, not last
      10 A6     D4     0D24  3026          CLRL    CXB$L_LINK(R6)                 ; Zero this CXB's link
  10 A1   56    D0     0D27  3027          MOVL    R6,CXB$L_LINK(R1)              ; Link it into end of chain
      0D A8   96    0D2B  3028          INCB    LSB$B_X_CXBACT(R8)             ; Account for it
                      0D2E  3029          ;
        3A A7   B5    0D2E  3030          TSTW    IRP$L_IOST1+2(R7)              ; Any data left ?
          03    12    0D31  3031          BNEQ    XMT_COPY                       ; If NEQ, yes
         F9F9   30    0D33  3032          BSBW    XMT_REQ_DONE_OK                ; Move request for completion
                      0D36  3033 XMT_COPY:
  57    10 A8   D0     0D36  3034          MOVL    LSB$L_X_PND(R8),R7             ; Get next IRP
          B9    13     0D3A  3035          BEQL    100$                           ; If EQL, none left
      05  5B    E8     0D3C  3036          BLBS    R11,30$                        ; If LBS, okay to goto IPL 2
B2 0C A7  1F    E1     0D3F  3037          BBC     #31,IRP$L_PID(R7),400$         ; If BC not ALTSTART, must
                      0D44  3038 30$:     ;
                      0D44  3039          ;
```

```
                     0D44  3040          ;    Get a free CXB.  Expand CXB list if needed and if possible.
                     0D44  3041          ;
                     0D44  3042          ;
 56   0118 D5   0F   0D44  3043          REMQUE  @XWB$Q_FREE_CXB(R5),R6          ; Get next CXB
            28   1C   0D49  3044          BVC     50$                             ; If VC, got one
            56   D4   0D4B  3045          CLRL    R6                              ; Init pointer
         0F A8   91   0D4D  3046          CMPB    LSB$B_X_CXBCNT(R8),-            ; Can we allocate another CXB ?
         0E A8                0D50  3047                  LSB$B_X_CXBQUO(R8)      ;
            A1   1E   0D52  3048          BGEQU   100$                            ; If GEQ, no
   51    42 A5   3C   0D54  3049          MOVZWL  XWB$W_REMSIZ(R5),R1             ; Get remote size
                      0D58  3050                                                  ;; trim upon ENT_RUN if needed
   51    5B A1   9E   0D58  3051          MOVAB    NSP$C_HSZ_DATA -               ; Add in overhead
                      0D5C  3052                   +TR3$C_HSZ_DATA -              ;
                      0D5C  3053                   +CXB$C_OVERHEAD(R1),R1         ;
 00000000'GF   16     0D5C  3054          JSB      G^EXE$ALONONPAGED              ; Allocate the buffer
         94   50     0D62  3055          BLBC     R0,200$                        ; If LBC then allocation failure
      56      52     0D65  3056          MOVL     R2,R6                          ; Setup CXB pointer
   0A A6      1B     0D68  3057          MOVB    #DYN$C_CXB,CXB$B_TYPE(R6)       ; Setup block type
   08 A6      51     0D6C  3058          MOVW     R1,CXB$W_SIZE(R6)              ; Setup the size
         0F A8   96   0D70  3059          INCB     LSB$B_X_CXBCNT(R8)             ; Account for CXB
                      0D73  3060 50$:     ;
                      0D73  3061          ;
                      0D73  3062          ;    Enter message type code.  Process 'bom' and 'eom' flags.
                      0D73  3063          ;
                      0D73  3064          ;
                      0D73  3065          ASSUME  NSP$C_MSG_DATA  EQ  0           ; 'Data' message type code for
                      0D73  3066          ASSUME  NSP$V_DATA_BOM  EQ  LSB$V_BOM
                      0D73  3067          ASSUME  NSP$V_DATA_EOM  EQ  LSB$V_EOM
                      0D73  3068                                                  ;
2B A8      DF 8F      0D73  3069          BICB3   #^C<LSB$M_BOM>,LSB$B_STS(R8),-  ; Enter message type code
         4E A6                0D78  3070                  CXB$B_X_NSPTYP(R6)      ;
      2B A8   20   8A   0D7A  3071          BICB    #LSB$M_BOM,LSB$B_STS(R8)        ; Preset next type code
      52    42 A5   3C   0D7E  3072          MOVZWL  XWB$W_REMSIZ(R5),R2             ; Get segment size
      3A A7      52   B1   0D82  3073          CMPW    R2,IRP$L_IOST1+2(R7)            ; More data left after this ?
                 12   1F   0D86  3074          BLSSU   70$                             ; If LSSU, more data left
      52    3A A7   3C   0D88  3075          MOVZWL  IRP$L_IOST1+2(R7),R2            ; Else, take it all
09 20 A7      08   E0   0D8C  3076          BBS     #IO$V_MULTIPLE,IRP$W_FUNC(R7),70$ ; If BS, not 'end of msg'
4E A6    40   8F   88   0D91  3077          BISB    #NSP$M_DATA_EOM,CXB$B_X_NSPTYP(R6) ; Set 'end of message flag'
   2B A8      20   88   0D96  3078          BISB    #LSB$M_BOM,LSB$B_STS(R8)        ; Preset next type code
                      0D9A  3079 70$:     ;
                      0D9A  3080          ;
                      0D9A  3081          ;    Update user VA descriptor in IRP, copy data into CXB
                      0D9A  3082          ;
                      0D9A  3083          ;
04 A6      57 A6   9E   0D9A  3084          MOVAB   CXB$T_X_DATA(R6),4(R6)          ; Setup destination pointer
   51      3C A7   D0   0D9F  3085          MOVL    IRP$L_IOST2(R7),R1              ; Get address of user data
         66      51   D0   0DA3  3086          MOVL    R1,(R6)                         ; Save user VA
   0C A6      52   B0   0DA6  3087          MOVW    R2,CXB$W_LENGTH(R6)             ; Save # user bytes in CXB
   3C A7      52   C0   0DAA  3088          ADDL    R2,IRP$L_IOST2(R7)              ; Update address
   3A A7      52   A2   0DAE  3089          SUBW    R2,IRP$L_IOST1+2(R7)            ; Consume bytes
      FF4A CF      9F   0DB2  3090          PUSHAB  XMT_COPYT                       ; Setup return address
                        0DB6  3091                                                  ; fall thru to COPY_DATA
                        0DB6  3092                  .DSABL  LSB
```

```
                                                D 7
                                ;3094
                                ;3095 COPY_DATA:
36 0C A7    1F    E0  0DB6 3096         BBS     #31,IRP$L_PID(R7),70$           ; If BS, then 'ALTSTART'
                      0DBB 3097         ;
                      0DBB 3098         ;
                      0DBB 3099         ;   Probe the user buffer.  The probing code relies on the fact that
                      0DBB 3100         ;   the first and last pages in the probe range are simultaneously
                      0DBB 3101         ;   probed -- hence we can probe two pages at a time.
                      0DBB 3102         ;
                      0DBB 3103         ;   Enter data into message.  Since we may be at IPL 2, it is possible
                      0DBB 3104         ;   that an NET$C_IPL event could cause the link to break and the
                      0DBB 3105         ;   IRP's to be cleaned up.  Therefore, the IRP cannot be referenced
                      0DBB 3106         ;   once we go below NET$C_IPL.
                      0DBB 3107         ;
                      0DBB 3108         ;
54 0B A7    02  00    EF  0DBB 3109         EXTZV   #0,#2,IRP$B_RMOD(R7),R4         ; Get request access mode
   53 FE00  8F    32  0DC1 3110         CVTWL   #-512,R3                        ; Set addition constant
   50 51    52    C1  0DC6 3111         ADDL3   R2,R1,R0                        ; Calc end of buffer
   51 01FF  8F    AA  0DCA 3112         BICW    #VA$M_BYTE,R1                   ; Must go to begining of page
                      0DCF 3113                                                 ; (since two pages worth of data
                      0DCF 3114                                                 ; could otherwise span 3 frames)
      50    51    C2  0DCF 3115         SUBL    R1,R0                           ; Calc # of bytes to probe
                      0DD2 3116         SETIPL  #IPL$_ASTDEL                    ; Allow paging
                      0DD5 3117         ;
08 2A A7    01    E1  0DD5 3118 30$:    BBC     #IRP$V_FUNC,IRP$W_STS(R7),50$   ; If BC, IO$_WRITELBLK
                      0DDA 3119                                                 ; (IRP$W_STS is remains valid
                      0DDA 3120                                                 ;  even if IRP has been sent to
                      0DDA 3121                                                 ;  IOPOST).
61    50    54    0D  0DDA 3122         PROBEW  R4,R0,(R1)                      ; Can user VA be written ?
                08  12  0DDE 3123         BNEQ    60$                             ; If NEQ, yes
                26  11  0DE0 3124         BRB     200$                            ; Report access violation
                      0DE2 3125         ;
61    50    54    0C  0DE2 3126 50$:    PROBER  R4,R0,(R1)                      ; Can user VA be read ?
                20  13  0DE6 3127         BEQL    200$                            ; If EQL no, report error
      51    53    C2  0DE8 3128 60$:    SUBL    R3,R1                           ; Update address of buffer
   50   6043    3E  0DEB 3129         MOVAW   (R0)[R3],R0                     ; Shrink total by 2 pages
                E4  14  0DEF 3130         BGTR    30$                             ; If GTR, more to probe
                      0DF1 3131         ;
04 B6    00 B6    52  28  0DF1 3132 70$:    MOVC3   R2,@(R6),@4(R6)                 ; Enter data
                      0DF7 3133         ;
                      0DF7 3134 100$:   SETIPL  #NET$C_IPL                      ; Go back to synchronizing IPL
   55    FF5C C8    9E  0DFA 3135         MOVAB   -XWB$T_DT(R8),R5                ; Recover XWB address
14 1C A5    07    E1  0DFF 3136         BBC     #XWB$V_FLG_SDT,XWB$W_FLG(R5),210$ ; If BC, not in RUN state
      50    01    90  0E04 3137         MOVB    #1,R0                           ; Say "success"
                05  0E07 3138         RSB                                     ; Done
                      0E08 3139
                      0E08 3140
                      0E08 3141
                      0E08 3142 200$:   ;
                      0E08 3143         ;
                      0E08 3144         ;   Access violations are fatal.  We have no choice but to break the
                      0E08 3145         ;   link since a portion of the user's buffer may have already been
                      0E08 3146         ;   transmitted.
                      0E08 3147         ;
                      0E08 3148         ;
                      0E08 3149         SETIPL  #NET$C_IPL                      ; Go back to synchronizing IPL
   55    FF5C C8    9E  0E0B 3150         MOVAB   -XWB$T_DT(R8),R5                ; Recover XWB address
```

```
                1C A5    01    A8  0E10  3151            BISW    #XWB$M_FLG_BREAK,XWB$W_FLG(R5)  ; Cause link to break
                         0C    DD  0E14  3152            PUSHL   #SS$_ACCVIO                     ; Store error status
                         02    11  0E16  3153            BRB     220$                            ; Continue
                                   0E18  3154                                                    ;
                         2C    DD  0E18  3155  210$:     PUSHL   #SS$_ABORT                      ; Store error state
                      50 56    D0  0E1A  3156  220$:     MOVL    R6,R0                           ; Prepare for deallocation
                         56    D4  0E1D  3157            CLRL    R6                              ; Clear former CXB pointer
                     F1DE'     30  0E1F  3158            BSBW    NET$DEALLOCATE                  ; Deallocate the block
                      50 8ED0      0E22  3159            POPL    R0                              ; Pickup error status
                         05        0E25  3160            RSB                                     ; Done
                                   0E26  3161
```

```
                                0E26  3163          .SBTTL  CLONE_RCV_CXB   - Clone a copy of a rcv'd CXB
                                0E26  3164 ;+
                                0E26  3165 ;
                                0E26  3166 ;  INPUTS:       R7      Scratch
                                0E26  3167 ;                R6      Current CXB address
                                0E26  3168 ;                R2      Bytes left in message
                                0E26  3169 ;                R1      Current message pointer
                                0E26  3170 ;                R0      Scratch
                                0E26  3171 ;
                                0E26  3172 ;  OUTPUTS:      R7      Old CXB address
                                0E26  3173 ;                R6      New CXB address if successful
                                0E26  3174 ;                        otherwise unchanged
                                0E26  3175 ;                R1      Updated message pointer if successful
                                0E26  3176 ;                        garbage otherwise
                                0E26  3177 ;                R0      Low bit set if successful
                                0E26  3178 ;                        Low bit cleared otherwise
                                0E26  3179 ;
                                0E26  3180 ;                All other registers are preserved
                                0E26  3181 ;
                                0E26  3182 ;-
                                0E26  3183 CLONE_RCV_CXB_1:                                     ;
      51    66    D0            0E26  3184          MOVL    (R6),R1                           ; Get pointer to data
                                0E29  3185 CLONE_RCV_CXB:                                       ;
                                0E29  3186          ;
                                0E29  3187          ;
                                0E29  3188          ;   If we cannot take the buffer, then we can only write in the
                                0E29  3189          ;   CXB$x_R_xxx fields since that is the "datalink" area and the rest
                                0E29  3190          ;   of the CXB is off limits to us.
                                0E29  3191          ;
                                0E29  3192          ;   NOTE: If this is a local-local link then this is the same CXB as
                                0E29  3193          ;         on some LSB CXB list and modifying anything before CXB$T_DLL
                                0E29  3194          ;         or after CXB$C_HEADER could be dangeous.  Remember that for
                                0E29  3195          ;         local-local links the flag is set in the CXB which prohibits
                                0E29  3196          ;         its being consumed here.
                                0E29  3197          ;
                                0E29  3198          ;
      013C  8F    BB            0E29  3199          PUSHR   #^M<R2,R3,R4,R5,R8>             ; Save regs
                                0E2D  3200                                                  ;
      57    56    D0            0E2D  3201          MOVL    R6,R7                           ; Copy CXB address
      67    51    D0            0E30  3202          MOVL    R1,(R7)                         ; Enter pointer to data
      52    52    3C            0E33  3203          MOVZWL  R2,R2                           ; Clear out high order bits
      52    04    C0            0E36  3204          ADDL    #CXB$C_TRAILER,R2               ; Add in required trailer
      51    52    C0            0E39  3205          ADDL    R2,R1                           ; Calculate end address
      51    57    C2            0E3C  3206          SUBL    R7,R1                           ; Get total used buffer size
      58    51    D0            0E3F  3207          MOVL    R1,R8                           ; Save a copy
 00000000'GF      16            0E42  3208          JSB     G^EXE$ALONONPAGED               ; Allocate buffer
        1B    50  E9            0E48  3209          BLBC    R0,10$                          ; If LBC then error
      56    52    D0            0E4B  3210          MOVL    R2,R6                           ; Setup new CXB address
66    67    58    28            0E4E  3211          MOVC3   R8,(R7),(R6)                    ; Copy relavent data
  08  A6    58    B0            0E52  3212          MOVW    R8,CXB$W_SIZE(R6)               ; Reset size -- corrupted by MOVC
51    67    57    C3            0E56  3213          SUBL3   R7,(R7),R1                      ; Get offset to data
      51    56    C0            0E5A  3214          ADDL    R6,R1                           ; Make it a pointer in new CXB
      66    51    D0            0E5D  3215          MOVL    R1,(R6)                         ; Store it
        38  A6    94            0E60  3216          CLRB    CXB$B_R_FLG(R6)                 ; Clear all flags
      50    01    D0            0E63  3217          MOVL    #1,R0                           ; Success
                                0E66  3218                                                  ;
      013C  8F    BA            0E66  3219 10$:     POPR    #^M<R2,R3,R4,R5,R8>             ; Restore regs
```

```
05  0E6A  3220          RSB                             ; Done
    0E6B  3221
    0E6B  3222
```

```
                     0E6B  3224
                     0E6B  3225 NET$DRAIN_R_IRPCXB::
    50   2C A3   D0  0E6B  3226 10$:     MOVL    IRP$L_SVAPTE(R3),R0                 ; Get attached CXB, if any
           0D    13  0E6F  3227          BEQL    100$                                ; If EQL, done
2C A3    10 A0   D0  0E71  3228          MOVL    CXB$L_LINK(R0),IRP$L_SVAPTE(R3)     ; Remove CXB from list
         05 A3   97  0E76  3229          DECB    IRP$B_CXBCNT(R3)                    ; Account for its loss
          F184'  30  0E79  3230          BSBW    NET$DEALLOCATE                      ; Deallocate the block
           ED    11  0E7C  3231          BRB     10$                                 ; Loop
                 05  0E7E  3232 100$:    RSB                                         ; Done
                     0E7F  3233
                     0E7F  3234
                     0E7F  3235 NET$DRAIN_R_LSBCXB:
                     0E7F  3236                                                      ;
    50   20 A8   D0  0E7F  3237 10$:     MOVL    LSB$L_R_CXB(R8),R0                  ; Get attached CXB, if any
           0D    13  0E83  3238          BEQL    100$                                ; If EQL, done
20 A8    10 A0   D0  0E85  3239          MOVL    CXB$L_LINK(R0),LSB$L_R_CXB(R8)      ; Remove CXB from list
         28 A8   97  0E8A  3240          DECB    LSB$B_R_CXBCNT(R8)                  ; Account for its loss
          F170'  30  0E8D  3241          BSBW    NET$DEALLOCATE                      ; Deallocate the block
           ED    11  0E90  3242          BRB     10$                                 ; Loop
                 05  0E92  3243 100$:    RSB                                         ; Done
                     0E93  3244
```

```
                                                I 7
                        .SBTTL  NSP$SOLICIT     - Solicit permission to transmit
                ;+
                ;
                ;  It is asssumed that XWB$V_STS_SOL has just be set prior to the call to this
                ;  routine.
                ;
                ;
                ;  INPUTS:      R5      XWB address
                ;               R4-R0   Scratch
                ;
                ;               R5-R0   Garbage
                ;
                ;               All other registers are preserved.
                ;
                ;-
                NSP$SOLICIT::                           ; Solicit xmit permission from Trasnport
                        ;
                        ;   Solicit permission from Transport to transmit a message.  Note that
                        ;   the request could suspend us indefinitely.  The call is made with:
                        ;
                        ;       R5      Fork block address.
                        ;               The FPC,FR3,FR4 fields are all scratch and must not
                        ;               be modified by while Transport owns the fork block.
                        ;       R4      Destination node address
                        ;       R3      I.d. of LPD to xmit over
                        ;               Zero if Transport is to choose the LPD
                        ;       R2      RCB address
                        ;       R1,R0   Scratch
                        ;
                        ;       (SP)    Return address of caller
                        ;      4(SP)    Return address of caller's caller
                        ;
                        ;
 52   30 A5   D0  0E93  20$:    MOVL    XWB$L_VCB(R5),R2                ; Get RCB address
 53   38 A5   3C  0E97          MOVZWL  XWB$W_PATH(R5),R3               ; Get path i.d. for xmt
 55   14 A5   9E  0E9B          MOVAB   XWB$Q_FORK(R5),R5               ; Switch to fork block context
      AE'AF   9F  0E9F          PUSHAB  B^QUICK_SOL                     ; Setup return address
54  FFEC'C5   9A  0EA2          MOVZBL  W^XWB$B_ADJ_INX-XWB$Q_FORK(R5),R4 ; Setup adjacency index
                  0EA7          BEQL    30$                             ; If EQL, none
                  0EA7          BRW     QRL$SOLICIT                     ; Call quick routing layer
 54   26 A5   3C  0EA7  30$:    MOVZWL  XWB$W_REMNOD-XWB$Q_FORK(R5),R4  ; Setup remote node address
       F152'  31  0EAB          BRW     TR$SOLICIT                      ; Call Transport
                  0EAE
                  0EAE  QUICK_SOL:
                        ;
                        ;   Return (or called) from Transport with:
                        ;
                        ;       R7,R6   Scratch
                        ;       R5      Fork block address
                        ;       R4      Scratch
                        ;       R3      Not available -- must be saved/restored
                        ;       R2      RCB address
                        ;       R1      Scratch
```

```
                                                                J 7
                                0EAE  3303          ;       R0      Low bit set if permission granted
                                0EAE  3304          ;               Low bit clear if permission denied
                                0EAE  3305          ;
                                0EAE  3306          ;
                                0EAE  3307          ;
                    0F08 8F     BB  0EAE  3308          PUSHR   #^M<R3,R8,R9,R10,R11>           ; Save regs
                                0EB2  3309          ;
                          5B    D4  0EB2  3310          CLRL    R11                             ; Say "can't go to IPL 2"
                       55   EC A5  9E  0EB4  3311          MOVAB   -XWB$Q_FORK(R5),R5              ; Switch to XWB context
                       0E   A5 04  AA  0EB8  3312          BICW    #XWB$M_STS_SOL,XWB$W_STS(R5)    ; Mark fork block idle
                 54 1C A5   10 00  EA  0EBC  3313          FFS     #0,#16,XWB$W_FLG(R5),R4         ; Find something to do
                    59  1263'CF    9E  0EC2  3314          MOVAB   W^NET$IO_STATUS,R9              ; Default I/O end-action routine
                                7B  10  0EC7  3315          BSBB    BLD_DISPATCH                    ; Dispatch to build message
                             5E 50  E9  0EC9  3316          BLBC    R0,200$                         ; If LBC then no msg was built
                                0ECC  3317          BUMP    L,NDC+NDC$L_PSN(R5)             ; Update "packets sent"
                                0ED7  3318          UPDATE  L,R1,NDC+NDC$L_BSN(R5)          ; Update "user bytes sent"
                                0EE3  3319          ;
                                0EE3  3320          ;
                                0EE3  3321          ;    Build the route header
                                0EE3  3322          ;
                                0EE3  3323          ;
                    52   0000'C5    D0  0EE3  3324          MOVL    XWB$L_PTR_RTHD(R5),R2           ; Get route-header pointer
                      51    4E A6  9E  0EE8  3325          MOVAB   CXB$B_X_NSPTYP(R6),R1           ; Setup pointer to msg NSP header
                      51    FC A2  C2  0EEC  3326          SUBL    -4(R2),R1                       ;
                    57   53   51    C3  0EF0  3327          SUBL3   R1,R3,R7                        ; Setup total message size
                                0EF4  3328          ;
                          23    BB  0EF4  3329          PUSHR   #^M<R0,R1,R5>                   ; Save regs
                 61   62    FC A2  28  0EF6  3330          MOVC3   -4(R2),(R2),(R1)                ; Move in the route-header
                          23    BA  0EFB  3331          POPR    #^M<R0,R1,R5>                   ; Save regs
                                0EFD  3332          ;
                    52   59    D0  0EFD  3333          MOVL    R9,R2                           ; Setup 'end-action' routine
                          08    12  0F00  3334          BNEQ    120$                            ; If NEQ then okay
                 52 1263'CF    9E  0F02  3335          MOVAB   W^NET$IO_STATUS,R2              ; Use standard status routine
                    61   08    88  0F07  3336          BISB    #TR3$M_RTFLG_RQR,(R1)           ;
                                0F0A  3337          ;
                    0F08 8F     BA  0F0A  3338  120$:   POPR    #^M<R3,R8,R9,R10,R11>           ; Restore status and saved regs
                                0F0E  3339          ;
                                0F0E  3340          ;
                                0F0E  3341          ;
                                0F0E  3342          ;    On return, the CXB and registers are setup as follows:
                                0F0E  3343          ;
                                0F0E  3344          ;
                                0F0E  3345          ;    +---------------+
                                0F0E  3346          ;    |   standard    |   11 bytes long.  CXB$L_FLINK and CXB$L_BLINK may
                                0F0E  3347          ;    |     VMS       |   be used by the Transport layer.  CXB$Q_SIZE
                                0F0E  3348          ;    | buffer header |   must be correct.  CXB$B_TYPE must be DYN$C_CXB.
                                0F0E  3349          ;    +---------------+
                                0F0E  3350          ;    |               |   Starts with CXB$B_CODE (byte 11) and continues
                                0F0E  3351          ;    |     ECL       |   to CXB$C_LENGTH.  This area is read-only to
                                0F0E  3352          ;    |   pure area   |   Transport and below.  It cannot even be
                                0F0E  3353          ;    |               |   saved/restored.
                                0F0E  3354          ;    +---------------+
                                0F0E  3355          ;    |   Datalink    |   Starts at CXB$C_LENGTH and is at least
                                0F0E  3356          ;    |     Layer     |   CXB$C_DLL bytes long.  Used by the datalink for
                                0F0E  3357          ;    |  impure area  |   protocol header or state information.
                                0F0E  3358          ;    +---------------+
                                0F0E  3359          ;    |               |
```

```
                                0F0E  3360          ;   .    body of    .   Must be quadword aligned and starting no sooner
                                0F0E  3361          ;   .    message    .   than CXB$C_LENGTH + CXB$C_DLL (= CXB$C_HEADER)
                                0F0E  3362          ;   |               |
                                0F0E  3363          ;   +---------------+
                                0F0E  3364          ;   |    Datalink  |   Used by the datalink layer for protocol (e.g.,
                                0F0E  3365          ;   |     Layer    |   checksum) or state information.  Must be at
                                0F0E  3366          ;   |  impure area |   least CXB$C_TRAILER in length.
                                0F0E  3367          ;   +---------------+
                                0F0E  3368          ;
                                0F0E  3369          ;
                                0F0E  3370          ;       R7      Size of message
                                0F0E  3371          ;       R6      CXB address
                                0F0E  3372          ;       R5      Garbage
                                0F0E  3373          ;       R4      0 if "quick solicit" not requested
                                0F0E  3374          ;               Else, pointer to request block (XWB fork block) with
                                0F0E  3375          ;               FRK$L_FPC  pointing to the "quick solicit" routine
                                0F0E  3376          ;       R3      IRP address -- unmodified from call
                                0F0E  3377          ;       R2      Address of End-action routine to call on I/O competion
                                0F0E  3378          ;       R1      Ptr to 1st byte in standard Phase III route-header
                                0F0E  3379          ;       R0      Low bit set   - if message is to be xmitted
                                0F0E  3380          ;               Low bit clear - if no message to xmit.  In this case
                                0F0E  3381          ;                               R7-R4,R2,R1 contain garbage.
                                0F0E  3382          ;
                                0F0E  3383          ;
                                0F0E  3384          ;
    54   1C A5    0A    00   EA  0F0E  3385          FFS     #0,#XWB$V_FLG_CLO+1,XWB$W_FLG(R5),R4  ; Get next work bit
         14 0000'CF    54   E1  0F14  3386          BBC     R4,W^NET$GL_WORKBITS,300$             ; If BC, no work needed
            22 0E A5   02   E2  0F1A  3387          BBSS    #XWB$V_STS_SOL,XWB$W_STS(R5),310$;    If BS, fork block in-use
         20 A5    FF8B CF   9E  0F1F  3388          MOVAB   W^QUICK_SOL,XWB$L_FPC(R5)             ; Setup "quick solicit" return
                54    14 A5   9E  0F25  3389          MOVAB   XWB$Q_FORK(R5),R4                     ; Say "quick solicit requested"
                               05  0F29  3390          RSB                                   ; Return to Transport
                                   0F2A  3391
                                   0F2A  3392
              0F08 8F    BA   0F2A  3393  200$:   POPR    #^M<R3,R8,R9,R10,R11>                 ; Restore status and saved regs
 0E 0E A5     03    E1   0F2E  3394  300$:   BBC     #XWB$V_STS_DIS,XWB$W_STS(R5),310$;    If BC, disconnect not pending
                                   0F33  3395                                                ;
                      50   DD   0F33  3396           PUSHL   R0                            ; Save R0
                    FOC8'  30   0F35  3397           BSBW    NET$CHK_X_IDLE                ; Ok to restart disc. sequence?
                  03 50    E9   0F38  3398           BLBC    R0,305$                       ; If LBC no, XWB is not idle
                    FOC2'  30   0F3B  3399           BSBW    NET$FORK                      ; Fork to resume disconnect
                                   0F3E  3400                                                ; (return with LBS in R0)
                      50 8ED0   0F3E  3401  305$:   POPL    R0                            ; Restore R0
                                   0F41  3402                                                ;
                      54   D4   0F41  3403  310$:   CLRL    R4                            ; Say "quick solicit not wanted"
                                05   0F43  3404           RSB                           ; Return to Transport
                                   0F44  3405
```

```
                     0F44  3407          .SBTTL  BLD_DISPATCH    - Dispatch to build message
                     0F44  3408 ;+
                     0F44  3409 ;
                     0F44  3410 ;  Dispatch with:
                     0F44  3411 ;
                     0F44  3412 ;       R9      Default end-action routine (NET$IO_STATUS) address
                     0F44  3413 ;       R8-R6   Scratch
                     0F44  3414 ;       R5      XWB address
                     0F44  3415 ;       R4      XWB$W_FLG work bit
                     0F44  3416 ;       R3-R1   Scratch
                     0F44  3417 ;       R0      LBS if permission granted to transmit
                     0F44  3418 ;               LBC if permission denied
                     0F44  3419 ;
                     0F44  3420 ;  On return:
                     0F44  3421 ;
                     0F44  3422 ;       R9      Address of NET$IO_STATUS or some other end-action routine
                     0F44  3423 ;       R6      Address of CXB containing message to be transmitted
                     0F44  3424 ;       R3      Address of first byte beyond the message text
                     0F44  3425 ;       R1      User bytes entered into message
                     0F44  3426 ;       R0      1 if message is to be xmitted,
                     0F44  3427 ;               0 otherwise
                     0F44  3428 ;
                     0F44  3429 ;       R8,R7,R4,R2  are clobbered, all others are unmodified.
                     0F44  3430 ;
                     0F44  3431 ;-
                     0F44  3432 BLD_DISPATCH:
1A 50   E9           0F44  3433          BLBC    R0,DENIED                       ; If LBC then we were denied
                     0F47  3434                                                  ; permission to transmit
                     0F47  3435          ;
                     0F47  3436          ;
                     0F47  3437          ;    We have been given permission to transmit.
                     0F47  3438          ;
                     0F47  3439          ;
                     0F47  3440          $DISPATCH  R4,-                         ; Case on work bit
                     0F47  3441          <-
                     0F47  3442              <XWB$V_FLG_CLO,     NET$RET_SLOT>,-
                     0F47  3443              <XWB$V_FLG_BREAK,   BREAK>,-
                     0F47  3444              <XWB$V_FLG_SCD,     BLD_CD>,-
                     0F47  3445              <XWB$V_FLG_SIACK,   BLD_LIACK>,-
                     0F47  3446              <XWB$V_FLG_SDACK,   BLD_DTACK>,-
                     0F47  3447              <XWB$V_FLG_SLI,     BLD_LI>,-
                     0F47  3448              <XWB$V_FLG_SDT,     BLD_DAT>,-
                     0F47  3449          >
41   11              0F5F  3450          BRB     NONE                            ; Continue
                     0F61  3451
                     0F61  3452 DENIED:  ;
                     0F61  3453          ;
                     0F61  3454          ;    Permission to Xmit has been denied
                     0F61  3455          ;
                     0F61  3456          ;
                     0F61  3457          $DISPATCH       R4,-                    ; Case on work bit
                     0F61  3458          <-
                     0F61  3459             <XWB$V_FLG_CLO,  NET$RET_SLOT>,-
                     0F61  3460          >
                     0F67  3461                                                  ; For all other bits, come here
                     0F67  3462          ;
                     0F67  3463          ;    If this is the first transmission of a CI then assume the node
```

```
                                                M 7
                     0F67  3464          ;    is unreachable
                     0F67  3465          ;
                     0F67  3466          ;
    1E A5    01  91  0F67  3467          CMPB   #XWB$C_STA_CIS,XWB$B_STA(R5)     ; Is a CI being sent ?
             11  12  0F6B  3468          BNEQ    50$                             ; If NEQ no
          52 A5  B5  0F6D  3469          TSTW    XWB$W_PROGRESS(R5)              ; Is this the 1st transmission ?
             0C  12  0F70  3470          BNEQ    50$                             ; If NEQ no
    44 A5    27  B0  0F72  3471          MOVW    #NET$C_DR_NOPATH,XWB$W_R_REASON(R5) ; Set up disconnect reason
    54 A5    01  B0  0F76  3472          MOVW   #1,XWB$B_RETRAN(R5)              ; Reduce msg retransmissions
    52 A5    01  B0  0F7A  3473          MOVW   #1,XWB$W_PROGRESS(R5)            ; Cause link to break
          0408  30  0F7E  3474  50$:      BSBW    UPD_PROGRESS                    ; Update the progress counter
04 0E A5    00  E0  0F81  3475          BBS    #XWB$V_STS_TID,XWB$W_STS(R5),140$; If BS then timer is owned
    50 A5    05  B0  0F86  3476          MOVW   #5,XWB$W_TIMER(R5)               ; Try again in 5 seconds
13 1C A5    00  E1  0F8A  3477  140$:    BBC    #XWB$V_FLG_BREAK,XWB$W_FLG(R5),NONE ; If BS, link is to be broken
                     0F8F  3478
                     0F8F  3479  BREAK:  ;
                     0F8F  3480          ;
                     0F8F  3481          ;    Generate an event to break the link
                     0F8F  3482          ;
                     0F8F  3483          ;
       03C2 8F   BB  0F8F  3484          PUSHR   #^M<R1,R6,R7,R8,R9>             ; Save regs
                     0F93  3485          ;
    1C A5    01   AA  0F93  3486          BICW    #XWB$M_FLG_BREAK,XWB$W_FLG(R5)  ; Prevent infinite looping
    57    00'8F   9A  0F97  3487          MOVZBL  #NETEVT$_DSCLNK,R7              ; Setup event code
          F062'   30  0F9B  3488          BSBW    NET$EVENT                       ; Process event
                     0F9E  3489          ;
       03C2 8F   BA  0F9E  3490          POPR    #^M<R1,R6,R7,R8,R9>             ; Restore regs
             50   7C  0FA2  3491  NONE:   CLRQ    R0                              ; Say ''nothing to send'' and
                     0FA4  3492          ;                                       ;     ''no user bytes in msg''
                  05  0FA4  3493          RSB                                     ; Done
                     0FA5  3494
```

```
                     OFA5  3496          .SBTTL  BLD_CD           - Build Connect/Disconnect messages
                     OFA5  3497          .SBTTL  BLD_CI           - Build a CI msg from XWB contents
                     OFA5  3498          .SBTTL  BLD_CA           - Build a CA msg from XWB contents
                     OFA5  3499          .SBTTL  BLD_CC           - Build a CC msg from XWB contents
                     OFA5  3500          .SBTTL  BLD_DI           - Build a DI msg from XWB contents
                     OFA5  3501          .SBTTL  BLD_DC           - Build A DC msg from XWB contents
                     OFA5  3502 ;++
                     OFA5  3503 ;
                     OFA5  3504 ;  The appropriate control message is contructed from the information
                     OFA5  3505 ;  in the XWB.
                     OFA5  3506 ;
                     OFA5  3507 ;
                     OFA5  3508 ;  INPUTS:       R9      Default end-action routine (NET$IO_STATUS) address
                     OFA5  3509 ;                R8-R6   Scratch
                     OFA5  3510 ;                R5      XWB address
                     OFA5  3511 ;                R4      XWB$M_FLG work bit
                     OFA5  3512 ;                R3-R0   Scratch
                     OFA5  3513 ;
                     OFA5  3514 ;  OUTPUTS:      R10     Preserved
                     OFA5  3515 ;                R9      Address of NET$IO_STATUS or NET$CSS_IOSTAT
                     OFA5  3516 ;                        Zero implies NET$IO_STATUS and also requests that
                     OFA5  3517 ;                        the "return to send bit" be set in the route-header
                     OFA5  3518 ;                R6      Address of CXB containing the message
                     OFA5  3519 ;                R5      Preserved
                     OFA5  3520 ;                R3      Pointer to first byte beyond the message
                     OFA5  3521 ;                R1      Number of user bytes entered into message
                     OFA5  3522 ;                R0      LBS if a message was constucted
                     OFA5  3523 ;                        LBC otherwise
                     OFA5  3524 ;
                     OFA5  3525 ;                R8,R7,R4,R2  are garbage
                     OFA5  3526 ;
                     OFA5  3527 ;
                     OFA5  3528 ;-
                     OFA5  3529 BLD_CD:                                          ; Build Connect or Disconnect message
 51    F0 8F    9A   OFA5  3530          MOVZBL  #NSP$C_HSZ_CD,R1              ; Setup maximum buffer size needed
         0268   30   OFA9  3531          BSBW    GET_XMT_BUF                   ; Get buffer
                     OFAC  3532                                                ; - no return on error
    0100 8F    AA   OFAC  3533          BICW    #XWB$M_FLG_SCD,-              ; Clear the bit which brought us here
      1C A5          OFB0  3534                   XWB$W_FLG(R5)                ;
                     OFB2  3535          $DISPATCH XWB$B_STA(R5),TYPE=B,-     ; Dispatch according to state
                     OFB2  3536          <-
                     OFB2  3537             <XWB$C_STA_CIS,  BLD_CI>,-        ; Build Connect Initate msg
                     OFB2  3538             <XWB$C_STA_CIR,  BLD_CA>,-        ; Build Connect Ack msg
                     OFB2  3539             <XWB$C_STA_CCS,  BLD_CC>,-        ; Build Connect Confirm msg
                     OFB2  3540             <XWB$C_STA_DIS,  BLD_DI>,-        ; Build Disconnect Initiate msg
                     OFB2  3541             <XWB$C_STA_DIR,  BLD_DC>,-        ; Build Disconnect Confirm msg
                     OFB2  3542          >
 50    56    D0   OFC5  3543          MOVL    R6,R0                         ; Else, setup for deallocation
         F035'  30   OFC8  3544          BSBW    NET$DEALLOCATE                ; Deallocate the block
           50   D4   OFCB  3545          CLRL    R0                            ; Indicate nothing to send
                05   OFCD  3546 10$:     RSB
                     OFCE  3547
                     OFCE  3548
                     OFCE  3549                  .ENABL  LSB
                     OFCE  3550
                     OFCE  3551 BLD_CI:                                       ; Build CI from XWB
           59   D4   OFCE  3552          CLRL    R9                            ; Request "return to sender"
```

```
                                                B 8
                     83   68 8F    90  0FD0  3553          MOVB    #NSP$C_MSG_CR,(R3)+     ; Enter msg type - CI "retransmit"
                          52 A5    B5  0FD4  3554          TSTW    XWB$W_PROGRESS(R5)      ; Is this the first transmission ?
                             04    12  0FD7  3555          BNEQ    5$                      ; If NEQ then no
                     FF A3   18    90  0FD9  3556          MOVB    #NSP$C_MSG_CI,-1(R3)    ; Else setup for initial CI
                                       0FDD  3557
                                       0FDD  3558 5$:      ASSUME  XWB$W_LOCLNK   EQ  2+XWB$W_REMLNK
                                       0FDD  3559          ASSUME  NSP$C_SRV_NFC  EQ  0
                                       0FDD  3560          ASSUME  NSP$V_INF_VER  EQ  0
                                       0FDD  3561
                        83   3C A5    D0  0FDD  3562          MOVL    XWB$W_REMLNK(R5),(R3)+  ; Enter dst,src link addresses
                    83   0201 8F    B0  0FE1  3563          MOVW    #NSP$C_SRV_REQ!-        ; Enter required SERVICE bits and
                                       0FE6  3564                    -                       ;  say "no flow control"
                                       0FE6  3565                    <NSP$C_INF_V40@8>,(R3)+ ;   and indicate Version 3.2
                        83   40 A5    B0  0FE6  3566          MOVW    XWB$W_LOCSIZ(R5),(R3)+  ; Enter rcv segment size
                    54   010C C5    D0  0FEA  3567          MOVL    XWB$L_ICB(R5),R4        ; Get ICB
                      51   28 A4    9E  0FEF  3568          MOVAB   ICB$B_RPRNAM(R4),R1     ; Get dst process name address
                             F00A'   30  0FF3  3569          BSBW    NET$MOV_USTR            ; Move string without the count field
                      51   14 A4    9E  0FF6  3570          MOVAB   ICB$B_LPRNAM(R4),R1     ; Get src process name address
                             F003'   30  0FFA  3571          BSBW    NET$MOV_USTR            ; Move string without the count field
                                53   DD  0FFD  3572          PUSHL   R3                      ; Save current output ptr
                                83   94  0FFF  3573          CLRB    (R3)+                   ; Assume no data or access info
                      51   3C A4    9E  1001  3574          MOVAB   ICB$B_ACCESS(R4),R1     ; Point to access info strings
                                61   95  1005  3575          TSTB    (R1)                    ; Are access strings null ?
                                07   13  1007  3576          BEQL    10$                     ; If EQL then null
                     00 BE      01   90  1009  3577          MOVB    #1,@(SP)                ; Flag 'access info present'
                           EFF0'   30  100D  3578          BSBW    NET$MOV_USTR            ; Move string without the count field
                         9E   02    88  1010  3579 10$:     BISB    #2,@(SP)+               ; Flag 'data present' - it may be null
                                       1013  3580          BUMP    W,NDC+NDC$W_CSN(R5)     ; Update "connects sent"
                                3A   11  101E  3581          BRB     30$                     ; Continue in common
                                       1020  3582
                                       1020  3583 BLD_CC:                                 ; Build the Connect Confirm message
                                02   E1  1020  3584          BBC     #XWB$V_PRO_PH2,-        ; If BC, not Phase II
                           05 5A A5        1022  3585                  XWB$B_PRO(R5),15$       ;
                     59   123A'CF   9E  1025  3586          MOVAB   W^NET$CCS_IOSTAT,R9     ; Setup I/O status return address
                          83   28    90  102A  3587 15$:     MOVB    #NSP$C_MSG_CC,(R3)+     ; Setup message type
                                       102D  3588
                                       102D  3589          ASSUME  XWB$W_LOCLNK   EQ  2+XWB$W_REMLNK
                                       102D  3590          ASSUME  NSP$C_SRV_NFC  EQ  0
                                       102D  3591          ASSUME  NSP$V_INF_VER  EQ  0
                                       102D  3592
                        83   3C A5    D0  102D  3593          MOVL    XWB$W_REMLNK(R5),(R3)+  ; Enter dst,src link addresses
                    83   0201 8F    B0  1031  3594          MOVW    #NSP$C_SRV_REQ!-        ; Enter required SERVICE bits and
                                       1036  3595                    -                       ;  say "no flow control"
                                       1036  3596                    <NSP$C_INF_V40@8>,(R3)+ ;   and indicate Version 3.2
                        83   40 A5    B0  1036  3597          MOVW    XWB$W_LOCSIZ(R5),(R3)+  ; Enter rcv segment size
                                1E   11  103A  3598          BRB     30$                     ; Move user data
                                       103C  3599
                                       103C  3600 BLD_CA:                                  ; Build CA from XWB contents
                           83   24    90  103C  3601          MOVB    #NSP$C_MSG_CA,(R3)+     ; Enter msg typ
                        83   3C A5    B0  103F  3602          MOVW    XWB$W_REMLNK(R5),(R3)+  ; Enter dst link address
                                06   11  1043  3603          BRB     25$                     ; Take "no data" exit
                                       1045  3604
                                       1045  3605 BLD_DC:                                  ; Build DC msg
                        83   48 8F    90  1045  3606          MOVB    #NSP$C_MSG_DC,(R3)+     ; Enter msg type
                                24   10  1049  3607          BSBB    BLD_DX_COMMON           ; Setup msg header
                                51   D4  104B  3608 25$:     CLRL    R1                      ; No user data
                                16   11  104D  3609          BRB     50$                     ; Exit in common
```

```
                        104F  3610
                        104F  3611 BLD_DI:                                  ; Build DI from XWB
          83    38  90  104F  3612          MOVB    #NSP$C_MSG_DI,(R3)+     ; Enter msg type
                1B  10  1052  3613          BSBB    BLD_DX_COMMON           ; Setup msg header
          57    01  AE  1054  3614          MNEGW   #1,R7                   ; Set timer i.d. (-1 => connect/discon)
             03D9  30  1057  3615          BSBW    RESET_TIMER             ; Set the retransmission timer
                        105A  3616                                          ; -- don't zero XWB$W_PROGRESS (it was
                        105A  3617                                          ;    zeroed as this state was entered)
    51    5B  A5  9E  105A  3618 30$:     MOVAB   XWB$B_DATA(R5),R1       ; Get address of optional data
             EF9F'  30  105E  3619          BSBW    NET$MOV_CSTR            ; Move data as a counted string
    51    5B  A5  9A  1061  3620          MOVZBL  XWB$B_DATA(R5),R1       ; Setup number of data bytes in message
          0100  8F  AA  1065  3621 50$:     BICW    #XWB$M_FLG_SCD,-        ; Clear flag which got us here
            1C  A5        1069  3622                  XWB$W_FLG(R5)       ;
          50    01  D0  106B  3623          MOVL    #1,R0                   ; Indicate that msg was built
                        106E  3624                                          ; R1 has # of optional data bytes
                    05  106E  3625          RSB                             ; Done
                        106F  3626
                        106F  3627                  .DSABL  LSB
                        106F  3628
                        106F  3629
                        106F  3630 BLD_DX_COMMON:                           ; Common disconnet msg building
                        106F  3631                  ;
                        106F  3632                  ;
                        106F  3633                  ;   If the partner is Phase II (V3.1) convert it to V3.2 so that
                        106F  3634                  ;   we get the benefit of timer support.   This ensures that broken
                        106F  3635                  ;   Phase II logical-links will always cleanup.
                        106F  3636                  ;
                        106F  3637                  ;
    5A  A5    04    8A  106F  3638          BICB    #XWB$M_PRO_PH2,XWB$B_PRO(R5)  ; Enable timer support
                        1073  3639 10$:     ;
                        1073  3640                  ;
                        1073  3641                  ;   Insert logical link address and disconnect reason code
                        1073  3642                  ;
                        1073  3643                  ;
                        1073  3644          ASSUME  XWB$W_LOCLNK  EQ  2+XWB$W_REMLNK
                        1073  3645
      83    3C  A5    D0  1073  3646          MOVL    XWB$W_REMLNK(R5),(R3)+  ; Enter dst,src link addresses
      63    46  A5    B0  1077  3647          MOVW    XWB$W_X_REASON(R5),(R3) ; Enter disconnect reason
    0000'8F       83    B1  107B  3648          CMPW    (R3)+,#NET$C_DR_INVALID ; Valid reason code ?
                    04    1F  1080  3649          BLSSU   20$                     ; If LSSU, okay
      FE  A3        09    B0  1082  3650          MOVW    #NET$C_DR_ABORT,-2(R3)  ; Else, jam in a default
                    05  1086  3651 20$:     RSB                             ; Done
                        1087  3652
```

```
                                1087  3654          .SBTTL  BLD_LIACK       - Build a INT/LS ACK message
                                1087  3655          .SBTTL  BLD_DTACK       - Build a DATA ACK message
                                1087  3656          .SBTTL  BLD_LI          - Build INT/LS message
                                1087  3657          .SBTTL  BLD_DAT         - Build DATA message
                                1087  3658 ;+
                                1087  3659 ;
                                1087  3660 ;  The appropriate message is built.  If the message to be built is an ACK and
                                1087  3661 ;  XWB$W_FLG indicates that there is a message which may be sent on the sub-
                                1087  3662 ;  channel then the ACK is sent 'piggy-backed' within that message - otherwise,
                                1087  3663 ;  an ACK message is built and sent.  Messages sent on either subchannel will
                                1087  3664 ;  always 'piggy-back' and ACK to help reduce retransmissions by the remote
                                1087  3665 ;  node in the lost message environment offered by Transport.
                                1087  3666 ;
                                1087  3667 ;
                                1087  3668 ;  INPUTS:      R9      Not used
                                1087  3669 ;               R8-R6   Scratch
                                1087  3670 ;               R5      XWB address
                                1087  3671 ;               R4      XWB$M_FLG work bit
                                1087  3672 ;               R3-R0   Scratch
                                1087  3673 ;
                                1087  3674 ;  OUTPUTS:     R6      Address of CXB containing the message
                                1087  3675 ;               R3      Pointer to first byte beyond the message
                                1087  3676 ;               R1      Number of user bytes entered into message
                                1087  3677 ;               R0      LBS if a message was constucted
                                1087  3678 ;                       LBC otherwise
                                1087  3679 ;
                                1087  3680 ;               R8,R7,R4,R2  are garbage.  All others are preserved.
                                1087  3681 ;
                                1087  3682 ;-
                                1087  3683 BLD_LIACK:                                        ; Build INT/LS ACK
03 1C A5    04    E0            1087  3684          BBS     #XWB$V_FLG_SLI,XWB$W_FLG(R5),BLD_LI ; Piggy-back ACK if possible
          00DF    31            108C  3685          BRW     BLD_ACK_LI                      ; Build header
                                108F  3686
                                108F  3687 BLD_LI:                                         ; Build INT or LS message
          00DC    30            108F  3688          BSBW    BLD_ACK_LI                      ; Build header
        41  50    E9            1092  3689          BLBC    R0,50$                          ; Failed if LBC
      83    57    B0            1095  3690          MOVW    R7,(R3)+                        ; Enter segment number
17 6C A5    05    E1            1098  3691          BBC     #NSP$V_FLW_INT,XWB$B_X_FLW(R5),10$ ; If BC then "Link Service"
                                109D  3692          ;
                                109D  3693          ;
                                109D  3694          ;   Xmit an INTERRUPT message
                                109D  3695          ;
                                109D  3696          ;
   4E A6    30    90            109D  3697          MOVB    #NSP$C_MSG_INT,CXB$B_X_NSPTYP(R6); Enter message type code
   50   10 A8    D0            10A1  3698          MOVL    LSB$L_X_PND(R8),R0              ; Get associated IRP
   51   32 A0    3C            10A5  3699          MOVZWL  IRP$W_BCNT(R0),R1               ; Setup number of user bytes
                                10A9  3700          ;
            32    BB            10A9  3701          PUSHR   #^M<R1,R4,R5>                   ; Save regs
63  38 A0    51    28            10AB  3702          MOVC3   R1,IRP$L_IOST1(R0),(R3)         ; Move data
            32    BA            10B0  3703          POPR    #^M<R1,R4,R5>                   ; Restore regs
                                10B2  3704          ;
            10    11            10B2  3705          BRB     30$                             ; Finish in common
                                10B4  3706 10$:      ;
                                10B4  3707          ;
                                10B4  3708          ;   Xmit LINK SERVICE (flow control/back-pressure) message
                                10B4  3709          ;
                                10B4  3710          ;
```

```
                4E A6    10   90  10B4  3711          MOVB   #NSP$C_MSG_LS,CXB$B_X_NSPTYP(R6)      ; Enter message type code
          83    6C A5    F0 8F  8B  10B8  3712          BICB3  #NSP$M_FLW_DRV,XWB$B_X_FLW(R5),(R3)+ ; Enter flow control mode
                83   6D A5    90  10BE  3713          MOVB    XWB$B_X_FLWCNT(R5),(R3)+               ; Enter flow control value
                     51   D4   10C2  3714          CLRL    R1                                     ; Setup # of user bytes
                          10C4  3715 30$:          ;
                          10C4  3716               ;
                          10C4  3717               ;    Common LS/INT message completion
                          10C4  3718               ;
                          10C4  3719               ;
                1C A5    10   AA  10C4  3720          BICW   #XWB$M_FLG_SLI,XWB$W_FLG(R5)           ; Clear work bit
                          10C8  3721                                                              ;
                          10C8  3722          ASSUME  XWB$V_STS_TID  EQ  0                        ;
                          10C8  3723                                                              ;
                  03 0E A5   E9  10C8  3724          BLBC    XWB$W_STS(R5),40$                     ; If LBC, timer is unowned
                       008A  31  10CC  3725          BRW     EX                                    ; Else, take common exit
                0E A5    02   A8  10CF  3726 40$:     BISW   #XWB$M_STS_TLI,XWB$W_STS(R5)          ; Mark INT/LI channel as owner
                       007C  31  10D3  3727          BRW     EX_T                                  ; Set the timer
                             05  10D6  3728 50$:     RSB                                           ; Done
                                 10D7  3729
                                 10D7  3730
                                 10D7  3731 BLD_DTACK:                                           ; Build DATA ACK
          1C A5      0060 8F   B3  10D7  3732          BITW   #XWB$M_FLG_WHGL!-                      ; Any wait conditions preventing
                                 10DD  3733                  XWB$M_FLG_WBP,XWB$W_FLG(R5)          ; DATA message xmission ?
                          11   13  10DD  3734          BEQL    BLD_DAT                               ; If not, piggy-back this ACK
                     51   07   9A  10DF  3735          MOVZBL #NSP$C_HSZ_ACK,R1                      ; Setup size of NSP message
                        012F   30  10E2  3736          BSBW    GET_XMT_BUF                           ; Get buffer for ACK
                                 10E5  3737                                                        ; - no return on error
                58    00A4 C5   9E  10E5  3738          MOVAB   XWB$T_DT(R5),R8                       ; Get subchannel block
                      83   04   90  10EA  3739          MOVB   #NSP$C_MSG_DTACK,(R3)+                 ; Enter message type
                        00AE   31  10ED  3740          BRW     BLD_ACK_DAT                           ; Build common header
                                 10F0  3741
                                 10F0  3742 BLD_DAT:                                             ; Build a DATA message
           58    00A4 C5   9E  10F0  3743          MOVAB   XWB$T_DT(R5),R8                       ; Get subchannel pointer
                        00C9   30  10F5  3744          BSBW    GET_XMT_CXB                           ; Get next CXB for transmission
                                 10F8  3745                                                        ; No return on error
                        00A3   30  10F8  3746          BSBW    BLD_ACK_DAT                           ; Build header past ACK field
             04 A8      57   B1  10FB  3747          CMPW    R7,LSB$W_HXS(R8)                      ; Highest sendable segment ?
                        1D   12  10FF  3748          BNEQ    70$                                   ; If NEQ no, there's more
             1C A5      20   A8  1101  3749          BISW   #XWB$M_FLG_WHGL,XWB$W_FLG(R5)          ; Set wait condition
                68      57   B1  1105  3750          CMPW    R7,LSB$W_LOX(R8)                      ; Is this the last seq queued ?
                        34   12  1108  3751          BNEQ    80$                                   ; If NEQ no, there's more
                                 110A  3752                                                        ;
          0E A8      0D A8   91  110A  3753          CMPB    LSB$B_X_CXBACT(R8),LSB$B_X_CXBQUO(R8) ; At our limit ?
                        2D   1E  110F  3754          BGEQU   80$                                   ; If GEQU, yes
          0C A8      0D A8   91  1111  3755          CMPB    LSB$B_X_CXBACT(R8),LSB$B_X_PKTWND(R8) ; Could we send more ?
                        26   1E  1116  3756          BGEQU   80$                                   ; If GEQU, no
                52      58   D0  1118  3757          MOVL    R8,R2                                 ; Setup LSB for call
                     FB9A   30  111B  3758          BSBW    FILL_XMT_CXBS                         ; Try to get more data
                                 111E  3759                                                        ;
          1B 5A A5     04   E1  111E  3760 70$:     BBC     #XWB$V_PRO_NAR,XWB$B_PRO(R5),80$      ; If BC, NAR option not allowed
                                 1123  3761               ;
                                 1123  3762               ;
                                 1123  3763               ;   We may request that the ACK be delayed in order to reduce the number
                                 1123  3764               ;   of messages being processed.  In order to get an overlap between the
                                 1123  3765               ;   the pipelined data stream and the returning ACK stream, we must ask
                                 1123  3766               ;   for an ACK half way (arbitrarily chosen) between into the maximum
                                 1123  3767               ;   pipeline currently allowed.
```

```
                        1123  3768          ;
                        1123  3769          ;
                        1123  3770 ;        SUBW3   LSB$W_HAR(R8),LSB$W_HXS(R8),R0  ; Get # of packets in the pipe
50   57   06 A8   A3    1123  3771          SUBW3   LSB$W_HAR(R8),R7,R0             ; Get # of packets in the pipe
       50   50   A0    1128  3772          ADDW    R0,R0                           ; Double it
     02 0C A8   E9    112B  3773          BLBC    LSB$B_X_PKTWND(R8),75$          ; If even okay
            50   96    112F  3774          INCB    R0                              ; Else adjust the threshold
     0C A8   50   91    1131  3775 75$:     CMPB    R0,LSB$B_X_PKTWND(R8)           ; Half that allowed?
            07   13    1135  3776          BEQL    80$                             ; If so, ask for an ACK
  83   4000 8F   A8    1137  3777          BISW    #NSP$M_SEQ_NAR,(R3)+            ; Suppress ACK for effeciency
            05   11    113C  3778          BRB     90$                             ; Continue
                        113E  3779          ;
                        113E  3780          ;
  83   4000 8F   AA    113E  3781 80$:     BICW    #NSP$M_SEQ_NAR,(R3)+            ; Make sure ACK is sent
                        1143  3782          ;
   51   0C A6   3C    1143  3783 90$:     MOVZWL  CXB$W_LENGTH(R6),R1             ; Setup number of user bytes
     53   51   C0    1147  3784          ADDL    R1,R3                           ; Advance R3 to end of message
                        114A  3785          ASSUME  XWB$V_STS_TID EQ 0              ;
                        114A  3786          ;
     0B 0E A5   E8    114A  3787          BLBS    XWB$W_STS(R5),EX                ; If LBS, timer already owned
   0E A5   02   AA    114E  3788          BICW    #XWB$M_STS_TLI,XWB$W_STS(R5)    ; Mark DATA channel as owner
                        1152  3789 EX_T:    ;
                        1152  3790          ;
                        1152  3791          ;    Common exits for sequenced messages
                        1152  3792          ;
                        1152  3793          ;
   48 A5   57   B0    1152  3794          MOVW    R7,XWB$W_TIM_ID(R5)             ; Setup timed segment's number
         02CE   30    1156  3795          BSBW    SET_TIMER_RUN                   ; Set the timer in RUN state
   02 A8   57   B0    1159  3796 EX:      MOVW    R7,LSB$W_LNX(R8)                ; This will be 'last no. sent'
50 08 A8   57   A3    115D  3797          SUBW3   R7,LSB$W_HAA(R8),R0             ; Gtr than 'high ACK acceptable'
   04 50   0B   E1    1162  3798          BBC     #11,R0,100$                     ; If BC then LNX leq HAA
   08 A8   57   B0    1166  3799          MOVW    R7,LSB$W_HAA(R8)                ; Else update HAA as well
      50   01   D0    116A  3800 100$:    MOVL    #1,R0                           ; Indicate message was built
                  05    116D  3801          RSB                                     ; Done
                        116E  3802          ;
                        116E  3803          ;
                        116E  3804 BLD_ACK_LI:                                      ; Build LS/INT common header
     58   00D4 C5   9E    116E  3805          MOVAB   XWB$T_LI(R5),R8                 ; Get subchannel block
                        1173  3806          ;
                        1173  3807          ASSUME  NSP$C_HSZ_ACK+2+16 LE IRP$C_LENGTH ; Use lookaside list
                        1173  3808          ;
      51   C4 8F   9A    1173  3809          MOVZBL  #IRP$C_LENGTH,R1                ; Setup size of NSP message
                        1177  3810          ;                                       ; large enough for Interrupt msg
            009A   30    1177  3811          BSBW    GET_XMT_BUF                     ; Get buffer for message
                        117A  3812          ;                                       ; - no return on error
   57   02 A8   01   A1    117A  3813          ADDW3   #1,LSB$W_LNX(R8),R7             ; Get next segment number
     57   F000 8F   AA    117F  3814          BICW    #^X<F000>,R7                    ; Mask off junk bits
                        1184  3815          ;
            83   14   90    1184  3816          MOVB    #NSP$C_MSG_LIACK,(R3)+          ; Set message type
      1C A5   04   AA    1187  3817          BICW    #XWB$M_FLG_SIACK,XWB$W_FLG(R5)  ; Need to send ACK is satisfied
   00 1C A5   54   E5    118B  3818          BBCC    R4,XWB$W_FLG(R5),10$            ; Clear flag that got us here
50 26 A8   F000 8F   AB    1190  3819 10$:     BICW3   #^X<F000>,LSB$W_HAX(R8),R0      ; Get ACK value
   17 0E A5   09   E5    1197  3820          BBCC    #XWB$V_STS_LINAK,XWB$W_STS(R5),ACK ; Br unless NAK is to be sent
            10   11    119C  3821          BRB     NAK                             ; Send as a NAK
                        119E  3822          ;
                        119E  3823          ;
                        119E  3824 BLD_ACK_DAT:                                     ; Build DATA ACK header
```

```
                1C A5    08   AA  119E  3825          BICW   #XWB$M_FLG_SDACK,XWB$W_FLG(R5)    ; Clear the work bit
          50   26 A8    F000 8F  AB  11A2  3826          BICW3  #^X<F000>,CSB$W_MAX(R8),R0        ; Get ACK value
                 05 0E A5    08   E5  11A9  3827          BBCC   #XWB$V_STS_DTNAK,XWB$W_STS(R5),ACK ; Br unless NAK is to be sent
                   50    1000 8F   A8  11AE  3828 NAK:     BISW   #NSP$M_ACK_NAK,R0                 ; Set the NAK flag
                                        11B3  3829 ACK:                                           ;
                                        11B3  3830          ASSUME  XWB$W_LOCLNK  EQ  2+XWB$W_REMLNK;
                                        11B3  3831                                                ;
                      83   3C A5    D0   11B3  3832          MOVL    XWB$W_REMLNK(R5),(R3)+           ; Enter link addresses
             83   50    8000 8F   A9   11B7  3833          BISW3  #NSP$M_ACK_VALID,R0,(R3)+         ; Enter ACK field
                   50    01   7D   11BD  3834          MOVQ   #1,R0                             ; R1=0  =>  no user bytes in msg
                                        11C0  3835                                                ; R0=1  =>  success, xmt message
                                  05   11C0  3836          RSB                                      ; Done
                                        11C1  3837
```

```
                                                     H 8
                                11C1  3839          .SBTTL  GET_XMT_CXB     - Get xmt CXB while in FDT context
                                11C1  3840 ;+
                                11C1  3841 ;
                                11C1  3842 ;  INPUTS:       R8      LSB address
                                11C1  3843 ;                R7,R6   Scratch
                                11C1  3844 ;                R5      XWB address
                                11C1  3845 ;                R3-R0   Scratch
                                11C1  3846 ;
                                11C1  3847 ;  OUTPUTS:      R8      Preserved
                                11C1  3848 ;                R7      Segment number
                                11C1  3849 ;                R6      CXB address if LBS in R0
                                11C1  3850 ;                R5,R4   Preserved
                                11C1  3851 ;                R3      Pointer to CXB$B_X_NSPTYP+1(R6)
                                11C1  3852 ;                R2,R1   Garbage
                                11C1  3853 ;                R0      Low bit set if return to caller's caller
                                11C1  3854 ;                        Else, garbage
                                11C1  3855 ;
                                11C1  3856 ;-
                                11C1  3857 GET_XMT_CXB:
    57   02 A8   01   A1        11C1  3858          ADDW3   #1,LSB$W_LNX(R8),R7                 ; Get next sequence number
57  57   0C   00   EF        11C6  3859          EXTZV   #0,#12,R7,R7                ; Strip off the junk
      56   08 A8   9E        11CB  3860          MOVAB   LSB$L_X_CXB-CXB$L_LINK(R8),R6       ; Init for CXB scan
      56   10 A6   D0        11CF  3861 10$:     MOVL    CXB$L_LINK(R6),R6           ; Get next CXB
                04   19        11D3  3862          BLSS    20$                         ; If LSS then got one
                                11D5  3863                                              ;
                                11D5  3864          BUG_CHECK  NETNOSTATE,FATAL         ; CXB list was empty
                                11D9  3865                                              ;
57   55 A6   0C   00   ED        11D9  3866 20$:     CMPZV   #0,#12,CXB$W_X_NSPSEQ(R6),R7        ; This it?
                EE   12        11DF  3867          BNEQ    10$                         ; If NEQ no, loop
                                11E1  3868          ;
                                11E1  3869          ;
                                11E1  3870          ;    If CXB$V_CD_XMT is currently set in CXB$B_CODE then it is already
                                11E1  3871          ;    queued to some datalink output queue --  since the datalink driver
                                11E1  3872          ;    may be using the datalink-header portion of the buffer, we must
                                11E1  3873          ;    clone a new CXB for transmission.
                                11E1  3874          ;
                                11E1  3875          ;
1F 0B A6   00   E3        11E1  3876          BBCS    #CXB$V_CD_XMT,CXB$B_CODE(R6),80$; If BC then buffer is free
   51   08 A6   3C        11E6  3877          MOVZWL  CXB$W_SIZE(R6),R1                   ; Get size of buffer
 00000000'GF   16        11EA  3878          JSB     G^EXE$ALONONPAGED                   ; Allocate the buffer
         1A 50   E9        11F0  3879          BLBC    R0,200$                     ; If LBC then allocation failure
                30   BB        11F3  3880                                              ;
                30   BB        11F3  3881          PUSHR   #^M<R4,R5>                  ; Save regs
      55   56   D0        11F5  3882          MOVL    R6,R5                       ; Copy old CXB pointer
      56   52   D0        11F8  3883          MOVL    R2,R6                       ; Setup new CXB pointer
66    65   51   28        11FB  3884          MOVC3   R1,(R5),(R6)                ; Clone the original message
   0B A6   01   90        11FF  3885          MOVB    #CXB$M_CD_XMT,CXB$B_CODE(R6)        ; Say "xmt in progress"
                30   BA        1203  3886          POPR    #^M<R4,R5>                  ; Recover XWB address
                                1205  3887                                              ;
   53   4F A6   9E        1205  3888 80$:     MOVAB   CXB$B_X_NSPTYP+1(R6),R3             ; Setup output pointer
      50   01   D0        1209  3889          MOVL    #1,R0                       ; Indicate sucess
                05   120C  3890          RSB                                 ; Done
                                120D  3891                                              ;
   1C A5   02   A8        120D  3892 200$:    BISW    #XWB$M_FLG_WBUF,XWB$W_FLG(R5)       ; Set wait flag
                8E   D5        1211  3893          TSTL    (SP)+                       ; Pop caller's address
                05   1213  3894          RSB                                 ; Return R0 to caller's caller
```

```
                                                I  8
                     1214  3896          .SBTTL  GET_XMT_BUF     - Get xmt buffer while in fork context
                     1214  3897 ;+
                     1214  3898 ;
                     1214  3899 ;  INPUTS:       R6      Scratch
                     1214  3900 ;                R3,R2   Scratch
                     1214  3901 ;                R1      Size of NSP portion of message
                     1214  3902 ;                R0      Scratch
                     1214  3903 ;
                     1214  3904 ;  OUTPUTS:      R6      Buffer (CXB) address
                     1214  3905 ;                R3      Pointer to message NSP area within buffer
                     1214  3906 ;                R2,R1   Garbage
                     1214  3907 ;                R0      Status
                     1214  3908 ;
                     1214  3909 ;                All other registers are unchanged
                     1214  3910 ;
                     1214  3911 ;                CXB$W_SIZE      Actual CXB block size
                     1214  3912 ;                CXB$B_TYPE      DYN$C_CXB
                     1214  3913 ;                CXB$B_CODE      CXB$M_CD_XMT
                     1214  3914 ;
                     1214  3915 ;
                     1214  3916 ;   If allocation failure, return is to caller's caller.
                     1214  3917 ;
                     1214  3918 ;
                     1214  3919 ;-
                     1214  3920 GET_XMT_BUF:                                      ; Get xmt buffer
 51    52 A1    9E   1214  3921          MOVAB    CXB$C_OVERHEAD -               ; Add in CXB
                     1218  3922                   +TR3$C_HSZ_DATA(R1),R1         ; + Transport msg overhead
00000000'GF    16   1218  3923          JSB      G^EXE$ALONONPAGED              ; Allocate the buffer
     12  50    E9   121E  3924          BLBC     R0,200$                        ; If LBC then allocation failure
     56    52   D0   1221  3925          MOVL     R2,R6                          ; Setup CXB pointer
                     1224  3926          ;
                     1224  3927          ;
                     1224  3928          ;    Fill in common CXB fields
                     1224  3929          ;
                     1224  3930          ;
                     1224  3931          ASSUME   CXB$B_CODE  EQ  1+CXB$B_TYPE
                     1224  3932
   011B  8F    B0    1224  3933          MOVW     #DYN$C_CXB+<1@<CXB$V_CD_XMT+8>>,-; Setup block type
      0A  A6         1228  3934                    CXB$B_TYPE(R6)                 ; ...and setup CXB$B_CODE
08 A6     51    B0   122A  3935          MOVW     R1,CXB$W_SIZE(R6)              ; Setup the size
53    4E  A6    9E   122E  3936          MOVAB    CXB$B_X_NSPTYP(R6),R3          ; Point to message area in buffer
                05   1232  3937 100$:    RSB                                     ; Return status in R0
                     1233  3938
1C A5     02    A8   1233  3939 200$:    BISW     #XWB$M_FLG_WBUF,XWB$W_FLG(R5)  ; Set wait flag
          8E    D5   1237  3940          TSTL     (SP)+                          ; Pop caller's address
                05   1239  3941          RSB                                     ; Return to caller's caller
                     123A  3942                                                  ; with LBC in R0
                     123A  3943
```

```
                                                J 8
                     123A  3945          .SBTTL  NET$IO_STATUS   - Receive xmit status from Transport layer
                     123A  3946          .SBTTL  NET$CCS_IOSTAT  - Receive xmit status for Phase II CC message
                     123A  3947 ;++
                     123A  3948 ;
                     123A  3949 ;  This routine is called by Transport to return transmit status to NSP.  The
                     123A  3950 ;  action is to deallocate the CXB if it is no longer in use.
                     123A  3951 ;
                     123A  3952 ;
                     123A  3953 ;  INPUTS:       R5      IRP address
                     123A  3954 ;                R4,R3   Scratch
                     123A  3955 ;                R2      RCB pointer
                     123A  3956 ;                R1      Scratch
                     123A  3957 ;                R0      CXB address (no longer attached to IRP)
                     123A  3958 ;
                     123A  3959 ;
                     123A  3960 ;  OUTPUTS:      R4,R3,R1,R0  are garbage.  All others are unchanged.
                     123A  3961 ;
                     123A  3962 ;
                     123A  3963 ;--
                     123A  3964 NET$CCS_IOSTAT::                             ; Receive status after sending a
                     123A  3965                                              ; Connect Confirm to a Phase II node
                     123A  3966
    0FE5 8F    BB    123A  3967          PUSHR   #^M<R0,R2,R5,R6,R7,R8,R9,R10,R11> ; Save regs
                     123E  3968                                              ;
       58   50     D0  123E  3969          MOVL    R0,R8                       ; Save temp copy of CXB address
       56   55     D0  1241  3970          MOVL    R5,R6                       ; Save temp copy of IRP address
 53    51 A8     3C  1244  3971          MOVZWL  CXB$W_X_NSPLOC(R8),R3       ; Get local link number
         EDB5'    30  1248  3972          BSBW    NET$XWB_LOCLNK              ; Find the XWB
        11 55     E8  124B  3973          BLBS    R5,20$                      ; If LBS not found
 57    00'8F     9A  124E  3974          MOVZBL  #NETEVT$_PH2CCS,R7          ; Setup event
   04  38 A6     E8  1252  3975          BLBS    IRP$L_IOST1(R6),10$         ; If LBS then no I/O error
 57    00'8F     9A  1256  3976          MOVZBL  #NETEVT$_DSCLNK,R7          ; Else indicate link failure
          5B     D4  125A  3977 10$:     CLRL    R11                         ; Say "can't go to IPL 2"
       EDA1'    30  125C  3978          BSBW    NET$EVENT                   ; Report the event
                    125F  3979                                              ;
    0FE5 8F    BA  125F  3980 20$:     POPR    #^M<R0,R2,R5,R6,R7,R8,R9,R10,R11> ; Restore all regs
                     1263  3981                                              ;
                     1263  3982                                              ; Fall thru to NET$IO_STATUS
                     1263  3983
                     1263  3984 NET$IO_STATUS::                             ; Receive xmit status
          01     8A  1263  3985          BICB    #CXB$M_CD_XMT,-             ; I/O no longer pending
    0B A0               1265  3986                  CXB$B_CODE(R0)              ;
          03     12  1267  3987          BNEQ    10$                         ; If NEQ then don't deallocate
       ED94'    30  1269  3988          BSBW    NET$DEALLOCATE              ; Deallocate the block
                05  126C  3989 10$:     RSB
                     126D  3990
                     126D  3991
```

```
                                126D  3993          .SBTTL  NET$TIMER       - Process NETDRIVER clock tick
                                126D  3994 ;+
                                126D  3995 ;
                                126D  3996 ;  *** t.b.s.***
                                126D  3997 ;
                                126D  3998 ;
                                126D  3999 ;
                                126D  4000 ;-
                                126D  4001 NET$TIMER::                                  ; ...tick...
                    0C30 8F     BB  126D  4002          PUSHR   #^M<R4,R5,R10,R11>      ; Save regs
                                1271  4003          DSBINT  UCB$B_FIPL(R4)          ; Raise to driver IPL
                         5B     D4  1278  4004          CLRL    R11                     ; Say "can't go to IPL 2"
                                127A  4005          ;
                                127A  4006          ;
                                127A  4007          ;    If mount count = 0 then we're shutting down the network -- stop
                                127A  4008          ;    the clock and signal NETACP by deactivating the local LPD.
                                127A  4009          ;
                                127A  4010          ;
                   52    34 A4    D0  127A  4011          MOVL    UCB$L_VCB(R4),R2        ; Get RCB
                         54 A2    B5  127E  4012          TSTW    RCB$W_MCOUNT(R2)        ; Still active
                            19    12  1281  4013          BNEQ    5$                      ; If NEQ then yes
                          ED7A'   30  1283  4014          BSBW    TR$KILL_LOC_LPD         ; Kill the local LPD
                         5C 50    E9  1286  4015          BLBC    R0,20$                  ; Br on error
                   0B A5    04    8A  1289  4016          BICB    #TQE$M_REPEAT,TQE$B_RQTYPE(R5) ; Stop the clock
                50    0000'CF    9E  128D  4017          MOVAB   W^NET$GL_OFF_DPTFLG,R0  ; Get address of offset
                50    0000'CF    C0  1292  4018          ADDL    W^NET$GL_OFF_DPTFLG,R0  ; Point to DPT$B_FLAG
                      60    04    8A  1297  4019          BICB    #DPT$M_NOUNLOAD,(R0)    ; Allow driver to be reloaded
                            49    11  129A  4020          BRB     20$                     ; Done
                                129C  4021 5$:          ;
                                129C  4022          ;
                                129C  4023          ;    Call the Transport layer timer service routine
                                129C  4024          ;
                                129C  4025          ;
                          ED61'   30  129C  4026          BSBW    TR$TIMER                ; Call Transport layer timer
                                129F  4027          ;
                                129F  4028          ;
                                129F  4029          ;    Process all NSP level clocks
                                129F  4030          ;
                                129F  4031          ;
                      54    24 A2    D0  129F  4032          MOVL    RCB$L_PTR_LTB(R2),R4    ; Get LTB
                            40    13  12A3  4033          BEQL    20$                     ; If EQL then none
                      55    E0 A4    9E  12A5  4034          MOVAB   -XWB$L_LINK+LTB$L_XWB(R4),R5 ; Prepare for scan
                      55    2C A5    D0  12A9  4035 10$:     MOVL    XWB$L_LINK(R5),R5       ; Get next XWB
                            36    13  12AD  4036          BEQL    20$                     ; If EQL then none left
                1C A5    0402 8F    B3  12AF  4037          BITW    #XWB$M_FLG_WBUF!-       ; Waiting for buffer
                                12B5  4038                      XWB$M_FLG_WDAT,XWB$W_FLG(R5) ;  or need to try for more data?
                            06    13  12B5  4039          BEQL    12$                     ; If EQL no
                          ED46'   30  12B7  4040          BSBW    NET$FORK                ; Service WBUF
                          F955    30  12BA  4041          BSBW    NET$QAST                ; Service WDAT
                   50    0118 D5    0F  12BD  4042 12$:     REMQUE  @XWB$Q_FREE_CXB(R5),R0  ; Get next idle buffer
                            07    1D  12C2  4043          BVS     13$                     ; If BS, none
                          ED39'   30  12C4  4044          BSBW    NET$DEALLOCATE          ; Deallocate the block
                         00B3 C5    97  12C7  4045          DECB    XWB$T_DT+LSB$B_X_CXBCNT(R5) ; Account for it
                           4A A5    B6  12CB  4046 13$:     INCW    XWB$W_ELAPSE(R5)        ; Track elapsed time
                           50 A5    B7  12CE  4047          DECW    XWB$W_TIMER(R5)         ; Update time left
                            D6    1A  12D1  4048          BGTRU   10$                     ; Br unless timeout
                05 0E A5    02    E1  12D3  4049          BBC     #XWB$V_STS_SOL,XWB$W_STS(R5),15$ ; If BC, not on solicit queue
```

```
                50 A5   B6   12D8  4050          INCW    XWB$W_TIMER(R5)                 ; Come back in another second
                   CC   11   12DB  4051          BRB     10$                             ; Done for now
             52    30 A5   D0   12DD  4052 15$:    MOVL    XWB$L_VCB(R5),R2                ; Setup RCB pointer
                      0A   10   12E1  4053          BSBB    TIMEOUT                         ; Process timeout
                      C4   11   12E3  4054          BRB     10$                             ; Loop
                           12E5  4055 20$:    ;
                           12E5  4056         ;
                           12E5  4057         ;    Return to the Exec
                           12E5  4058         ;
                           12E5  4059         ;
                           12E5  4060         ENBINT                                  ; Restore IPL
                     0C30 8F   BA   12E8  4061          POPR    #^M<R4,R5,R10,R11>              ; Restore context
                           05   12EC  4062          RSB
                           12ED  4063
                           12ED  4064
                           12ED  4065                  .ENABL LSB
                           12ED  4066
                           12ED  4067 TIMEOUT:
                           12ED  4068          $DISPATCH TYPE=B,XWB$B_STA(R5),-        ; Dispatch on link state
                           12ED  4069          <-                                      ;
                           12ED  4070              <XWB$C_STA_RUN, T_O_RUN>,-          ; RUN state
                           12ED  4071              <XWB$C_STA_CIS, T_O_CI>,-           ; Connect Initiate Sending state
                           12ED  4072              <XWB$C_STA_CCS, T_O_CC>,-           ; Connect Confirm Sending state
                           12ED  4073              <XWB$C_STA_DIS, T_O_DI>,-           ; Disconnect Init Sending state
                           12ED  4074          >                                       ; else,  fall thru
                      64   11   1300  4075          BRB     70$                             ; Continue
                           1302  4076
                           1302  4077 T_O_CI:                                         ; Timeout xmting CI msg
                           1302  4078 T_O_CC:                                         ; Timeout xmting CC msg
                           1302  4079 T_O_DI:                                         ; Timeout xmting DI msg
           1C A5   0100 8F   A8   1302  4080          BISW    #XWB$M_FLG_SCD,XWB$W_FLG(R5)    ; Set 'send Connect/disconnect
                      5C   11   1308  4081          BRB     70$                             ; Continue
                           130A  4082
                           130A  4083 T_O_RUN:;
                           130A  4084         ;
                           130A  4085         ;    Force a retransmission of all unACKed messages.  If the inactivity
                           130A  4086         ;    timer has expired but there are no outstanding ACKs, then send a
                           130A  4087         ;    harmless flow control message, which requires an ACK, to test the
                           130A  4088         ;    viability of the link.
                           130A  4089         ;
                           130A  4090         ;    Phase II logical-link do not timeout waiting for an ACK, but a
                           130A  4091         ;    message should still be send every "inactivity timer" interval in
                           130A  4092         ;    order to make sure the other side is still up.  If the other side
                           130A  4093         ;    is not up then the "progress" count on the XWB will reach its limit
                           130A  4094         ;    and the link will break.  If the the other end of the logical-link
                           130A  4095         ;    is gone but its node is up (e.g., other node crashes and reboots)
                           130A  4096         ;    then when it receives this message it will send a Disconnect
                           130A  4097         ;    Confirm as a response -- thus also breaking the link.
                           130A  4098         ;
                           130A  4099         ;
        31 5A A5   02   E0   130A  4100          BBS     #XWB$V_PRO_PH2,XWB$B_PRO(R5),60$ ; If BS, Phase II
                           130F  4101                                                  ;
                           130F  4102          ASSUME  XWB$V_STS_TID  EQ  0            ;
              2D 0E A5   E9   130F  4103          BLBC    XWB$W_STS(R5),60$               ; If LBC then inactivity timer
        18 0E A5   01   E0   1313  4104          BBS     #XWB$V_STS_TLI,XWB$W_STS(R5),50$ ; If BS, LI subchannel owns timer
                           1318  4105         ;
                           1318  4106         ;
```

```
                     1318  4107          ;    Timeout on DATA subchannel.  Reset LNX and shrink the transmit-
                     1318  4108          ;    packet-window.
                     1318  4109          ;
                     1318  4110          ;
 52   00A4 C5   9E   1318  4111          MOVAB   XWB$T_DT(R5),R2                 ; Setup LSB pointer
02 A2   06 A2   B0   131D  4112          MOVW    LSB$W_HAR(R2),LSB$W_LNX(R2)     ; Rexmt all unACKed segs
                     1322  4113                                                  ; ...and enforce LSB Rule 2b.
        F561    30   1322  4114          BSBW    SHRINK_XPW                      ; Shrink the xmt-packet-window
                     1325  4115                                                  ; - clobbers R0-R4.  May pass
                     1325  4116                                                  ;   off timer to a different
                     1325  4117                                                  ;   message
     17 0E A5   E9   1325  4118          BLBC    XWB$W_STS(R5),60$               ; If LBC, timer is now unowned
           01   E0   1329  4119          BBS    #XWB$V_STS_TLI,-                 ; If BS, timer given to LI
     45 0E A5        132B  4120                  XWB$W_STS(R5),80$               ; subchannel
           36   11   132E  4121          BRB     70$                             ; Else, update PROGRESS even if
                     1330  4122                                                  ; new XWB$W_TIM_ID value
                     1330  4123 50$:     ;
                     1330  4124          ;
                     1330  4125          ;    Timeout on LI subchannel
                     1330  4126          ;
                     1330  4127          ;
 52   00D4 C5   9E   1330  4128          MOVAB   XWB$T_LI(R5),R2                 ; Get LI LSB
02 A2   06 A2   B0   1335  4129          MOVW    LSB$W_HAR(R2),LSB$W_LNX(R2)     ; Rexmt all unACKed segs
                     133A  4130                                                  ; ...and enforce LSB Rule 2b.
   1C A5   10   A8   133A  4131          BISW   #XWB$M_FLG_SLI,XWB$W_FLG(R5)     ; Set 'send LI' flag
           26   11   133E  4132          BRB     70$                             ; Continue
                     1340  4133 60$:     ;
                     1340  4134          ;
                     1340  4135          ;    Cause (possibly null) flow control message to be sent in order to
                     1340  4136          ;    cause the partner node to send and ACK.  This is done to make sure
                     1340  4137          ;    that the partner node is still there.
                     1340  4138          ;
                     1340  4139          ;
  50 A5   4C A5   B0   1340  4140          MOVW    XWB$W_TIM_INACT(R5),XWB$W_TIMER(R5) ; Reset timer
1C A5   4000 8F   A8   1345  4141          BISW   #XWB$M_FLG_SDFL,XWB$W_FLG(R5)    ; Schedule flow ctl msg
                     134B  4142          ;
                     134B  4143          ;
                     134B  4144          ;    If the XWB$L_PID field is zero, then there is no current owner of
                     134B  4145          ;    this link and we are in the RUN state waiting to transmit the
                     134B  4146          ;    message currently committed to the pipeline.  Hence, make sure that
                     134B  4147          ;    backpressure relaxed in order to avoid deadlock -- e.g., if both
                     134B  4148          ;    ends of the link were in this state we would have deadlock.
                     134B  4149          ;
                     134B  4150          ;    The reason both relaxing backpressure avoids the deadlock is that
                     134B  4151          ;    receiving a message that we cannot buffer while in the RUN state
                     134B  4152          ;    with XWB$L_PID equal to zero will cause the link to be marked for
                     134B  4153          ;    disconnect (see routine BACK_PRESSURE).
                     134B  4154          ;
                     134B  4155          ;
        34 A5   D5   134B  4156          TSTL    XWB$L_PID(R5)                   ; Any owner process
           23   12   134E  4157          BNEQ    80$                             ; If NEQ yes
     00C4 C5   D5   1350  4158          TSTL    XWB$T_DT+LSB$L_R_CXB(R5)        ; Any data currently
           03   13   1354  4159          BEQL    65$                             ; If EQL then none
        005B    30   1356  4160          BSBW    NET$MARK_LINK                   ; Cause link to break
 15 0E A5   06   E1   1359  4161 65$:     BBC    #XWB$V_STS_RBP,XWB$W_STS(R5),80$; If BC then no backpressure
1C A5   0800 8F   A8   135E  4162          BISW   #XWB$M_FLG_TBPR,XWB$W_FLG(R5)    ; Relax (toggle) backpressure
           0D   11   1364  4163          BRB     80$                             ; Continue
```

```
                                                N 8
                        1366  4164 70$:     ;
                        1366  4165           ;
                        1366  4166           ;    Common timeout processing
                        1366  4167           ;
                        1366  4168           ;
                        1366  4169           BUMP    W,NDC+NDC$W_RTO(R5)                 ; Account for response timeout
                  16    1371  4170           BSBB    UPD_PROGRESS                        ; Update progress count
              EC8A'  30 1373  4171 80$:      BSBW    NET$FORK                            ; Fork to do new work
                     05 1376  4172 100$:     RSB                                         ; Done
                        1377  4173
                        1377  4174                   .DSABL  LSB
                        1377  4175
                        1377  4176
                        1377  4177
02 A0   06 A0   B1      1377  4178 RESET:    CMPW    LSB$W_HAR(R0),LSB$W_LNX(R0)         ; Anything to rexmt ?
                  0A 13 137C  4179           BEQL    10$                                 ; If EQL no
02 A0   06 A0   B0      137E  4180           MOVW    LSB$W_HAR(R0),LSB$W_LNX(R0)         ; Rexmt all unACKed segs
                        1383  4181                                                       ; ...and enforce LSB Rule 2b.
                  50 96 1383  4182           INCB    R0                                  ; Indicate this LSB was reset
                51   01 90 1385  4183           MOVB    #1,R1                               ; Set retransmit flag
                     05 1388  4184 10$:      RSB                                         ; Done
                        1389  4185
                        1389  4186
                        1389  4187 UPD_PROGRESS:                                        ; Decrement progress count
                  52 A5 B6 1389  4188           INCW    XWB$W_PROGRESS(R5)                  ; Account for lack of progress
                  52 A5 B1 138C  4189           CMPW    XWB$W_PROGRESS(R5),-                ; Has it grown too large ?
                  54 A5    138F  4190                   XWB$W_RETRAN(R5)                    ;
                     1E 1F 1391  4191           BLSSU   20$                                 ; If LSSU then okay
                     1F 10 1393  4192           BSBB    NET$MARK_LINK                       ; Mark link for break
             0E A5   20 A8 1395  4193           BISW    #XWB$M_STS_TMO,XWB$W_STS(R5)        ; Indicate timeout
                0000'8F B1 1399  4194           CMPW    #NET$C_DR_INVALID,-                 ;
                  44 A5    139D  4195                   XWB$W_R_REASON(R5)                  ; Reason code set yet?
                     04 12 139F  4196           BNEQ    10$                                 ; If NEQ then yes
                     27 B0 13A1  4197           MOVW    #NET$C_DR_NOPATH,-                  ; Setup local reason code
                  44 A5    13A3  4198                   XWB$W_R_REASON(R5)                  ;
                0000'8F B1 13A5  4199 10$:     CMPW    #NET$C_DR_INVALID,-                 ;
                  46 A5    13A9  4200                   XWB$W_X_REASON(R5)                  ; Remote reason be set yet?
                     04 12 13AB  4201           BNEQ    20$                                 ; If NEQ then yes
                     26 B0 13AD  4202           MOVW    #NET$C_DR_EXIT,-                    ; Setup code to send to partner
                  46 A5    13AF  4203                   XWB$W_X_REASON(R5)                  ;
                   007F 31 13B1  4204 20$:      BRW     RESET_TIMER                         ; Set NSP timer
                        13B4  4205
                        13B4  4206
                        13B4  4207 NET$MARK_LINK::                                      ; Mark the link for break
0E A5     01 AA 13B4  4208           BICW    #XWB$M_STS_TID,XWB$W_STS(R5)        ; Free-up the timer
1C A5     01 A8 13B8  4209           BISW    #XWB$M_FLG_BREAK,XWB$W_FLG(R5)      ; Mark link for abort
              EC41' 30 13BC  4210           BSBW    NET$FORK                            ; Fork to do new work
                    05 13BF  4211           RSB                                         ; Done
                       13C0  4212
```

```
                                13C0  4214          .SBTTL  TIMED_SEG_ACKED - Timed segment has been ACK'd
                                13C0  4215
                                13C0  4216                  .ENABL  LSB
                                13C0  4217
                                13C0  4218  TIMED_SEG_ACKED:                          ; Timed segment has been ACK'd
                                13C0  4219                  ;
                                13C0  4220                  ;
                                13C0  4221                  ;   INPUTS:     R5  XWB pointer
                                13C0  4222                  ;               R0  Scratch
                                13C0  4223                  ;
                                13C0  4224                  ;   OUTPUTS:    R0  Garbage
                                13C0  4225                  ;
                                13C0  4226                  ;               All other regs are unchanged
                                13C0  4227                  ;
                                13C0  4228                  ;
                                13C0  4229                  ;   Update delay estimate as a function of the former delay, the new
                                13C0  4230                  ;   round trip time (delta) and the 'weight' parameter (this value is
                                13C0  4231                  ;   store after being incremented by one).  The following shows the
                                13C0  4232                  ;   derivation of the formula used to compute 'delay'.
                                13C0  4233                  ;
                                13C0  4234                  ;    delay = ((delay*weight)+delta)/(weight+1)
                                13C0  4235                  ;    delay = delay + ((delta-delay)/(weight+1))
                                13C0  4236                  ;
                                13C0  4237                  ;    elapse = delta - delay    ('elapse' is biased to minus 'delay'
                                13C0  4238                  ;                              (when the msg was first sent and
                                13C0  4239                  ;                              (incremented each clock tick - it may
                                13C0  4240                  ;                              (be negative
                                13C0  4241                  ;
                                13C0  4242                  ;    delay = delay + elapse/(weight+1)
                                13C0  4243                  ;
                                13C0  4244                  ;
 50   4A A5   B0                13C0  4245                  MOVW    XWB$W_ELAPSE(R5),R0     ; Get elapsed time
          0F   18                13C4  4246                  BGEQ    5$                      ; If GEQ then arrived late
     50   50   AE                13C6  4247                  MNEGW   R0,R0                   ; Convert to positive number
 50   58 A5   A6                13C9  4248                  DIVW    XWB$W_DLY_WGHT(R5),R0   ; Get weighted adjustment
          50   B6                13CD  4249                  INCW    R0                      ; Ensure minimum of 1 sec change
                                 13CF  4250                                                  ; to allow for loss of fractional part
4E A5     50   A2                13CF  4251                  SUBW    R0,XWB$W_DELAY(R5)      ; Get new delay value
          16   11                13D3  4252                  BRB     25$                     ;
50   58 A5   A6                13D5  4253  5$:             DIVW    XWB$W_DLY_WGHT(R5),R0   ; Get weighted adjustment
          50   B6                13D9  4254                  INCW    R0                      ; Ensure minimum of 1 sec change
                                 13DB  4255                                                  ; to allow for loss of fractional part
50   4E A5   A0                13DB  4256                  ADDW    XWB$W_DELAY(R5),R0      ; Get new delay value
     14   50   B1                13DF  4257                  CMPW    R0,#NSP$C_MAX_DELAY     ; Compare  against max delay allowed
          03   1B                13E2  4258                  BLEQU   20$                     ; If LEQ (unsigned) then okay
     50   14   3C                13E4  4259                  MOVZWL  #NSP$C_MAX_DELAY,R0     ; Else use the max delay allowed
4E A5     50   B0                13E7  4260  20$:           MOVW    R0,XWB$W_DELAY(R5)      ; Update the delay
          04   14                13EB  4261  25$:           BGTR    30$                     ; If GTR (signed) then okay
4E A5     01   B0                13ED  4262                  MOVW    #1,XWB$W_DELAY(R5)      ; Use 1 sec as minimum delay
                                 13F1  4263  30$:                                            ; Fall thru
                                 13F1  4264  CANCEL_TIMER:                                   ;
                                 13F1  4265                  ;
                                 13F1  4266                  ;
                                 13F1  4267                  ;   See if an already xmitted segment is waiting to be ACK'd.
                                 13F1  4268                  ;
                                 13F1  4269                  ;   On the LI sub-channel, since neither negative flow control or
                                 13F1  4270                  ;   backpressure are allowed, this is merely a matter of checking
```

```
                                13F1  4271          ;    for HAR < LNX.
                                13F1  4272          ;
                                13F1  4273          ;    On the DATA sub-channel, since negative flow control and back-
                                13F1  4274          ;    pressure are allowed, it is necessary to check for  HAR < LNX,
                                13F1  4275          ;    and that the link is not backpressured off.
                                13F1  4276          ;
                                13F1  4277          ;
                                13F1  4278          ;    NOTE:  LSB Rule 2a + 2b imply that HAR is always leq LNX.  This
                                13F1  4279          ;           fact is used below to avoid messy 12 bit comparisons
                                13F1  4280          ;
                                13F1  4281          ;
                                13F1  4282          ;
      0E A5    03    AA        13F1  4283          BICW   #XWB$M_STS_TID!-                     ; Timer is unowned and init
                                13F5  4284                  XWB$M_STS_TLI,XWB$W_STS(R5)     ; subchannel flag to known value
   50   00A4 C5    9E        13F5  4285          MOVAB   XWB$T_DT(R5),R0                     ; Get DATA sub-channel LSB
 02 A0    06 A0    B1        13FA  4286          CMPW    LSB$W_HAR(R0),LSB$W_LNX(R0)         ; Anything sent but unACK'd
                05  13        13FF  4287          BEQL    40$                                 ; If EQL then no
 15 1C A5    06    E1        1401  4288          BBC    #XWB$V_FLG_WBP,XWB$W_FLG(R5),60$     ; If BC then no backpressure
   50   00D4 C5    9E        1406  4289 40$:     MOVAB   XWB$T_LI(R5),R0                     ; Get LI sub-channel LSB
          4C A5    B0        140B  4290          MOVW    XWB$W_TIM_INACT(R5),-               ; Setup inactivity timer
          50 A5              140E  4291                  XWB$W_TIMER(R5)                     ; assuming nothing to send
 02 A0    06 A0    B1        1410  4292          CMPW    LSB$W_HAR(R0),LSB$W_LNX(R0)         ; Anything sent but unACK'd
                33  13        1415  4293          BEQL    100$                                ; If EQL, no
      0E A5    02    A8        1417  4294          BISW   #XWB$M_STS_TLI,XWB$W_STS(R5)         ; Timer owner is LI subchannel
                               141B  4295          ;
                               141B  4296          ;
                               141B  4297          ;    Hand off the timer to the next un-ACKed segment
                               141B  4298          ;
                               141B  4299          ;
   50   06 A0    01    A1     141B  4300 60$:     ADDW3  #1,LSB$W_HAR(R0),R0                  ; Get next unACKed seg number
48 A5   50   F000 8F    AB     1420  4301          BICW3  #^X<F000>,R0,XWB$W_TIM_ID(R5)        ; Setup timed segment number
                               1427  4302          ;
                               1427  4303 SET_TIMER_RUN:                                     ; Set timer while in RUN state
      0E A5    01    A8        1427  4304          BISW   #XWB$M_STS_TID,XWB$W_STS(R5)         ; Claim ownership of timer
   4A A5    4E A5    AE        142B  4305          MNEGW   XWB$W_DELAY(R5),XWB$W_ELAPSE(R5)    ; Bias the elapsed time timer
          52 A5    B4        1430  4306          CLRW    XWB$W_PROGRESS(R5)                  ; Init PROGRESS
                               1433  4307          ;
                               1433  4308          ;
                               1433  4309 NET$RESET_TIMER::                                  ; Reset logical-link timer
                               1433  4310 RESET_TIMER:                                       ; Reset logical-link timer
                               1433  4311          ASSUME  NSP$C_MAX_DELAY  LT  <^X7FFF>
                               1433  4312          ;
          56 A5    A5        1433  4313          MULW3   XWB$W_DLY_FACT(R5),-                ;
   50   4E A5              1436  4314                  XWB$W_DELAY(R5),R0                  ; Get timer value
                05  1D        1439  4315          BVS     80$                                 ; If BS then overflow
        14   50    B1        143B  4316          CMPW    R0,#NSP$C_MAX_DELAY                 ; Greater than max ?
                03  1B        143E  4317          BLEQU   90$                                 ; If LEQU no, its okay
        50   14    B0        1440  4318 80$:     MOVW   #NSP$C_MAX_DELAY,R0                  ; Use max delay
50 A5   50   01    A1        1443  4319 90$:     ADDW3  #1,R0,XWB$W_TIMER(R5)                ; Set timer (+1 for clock skew)
                F6  15        1448  4320          BLEQ    80$                                 ; If LEQ (signed) then overflow
                05  144A  4321 100$:    RSB                                                 ; Done
                               144B  4322          ;
                               144B  4323          ;
                               144B  4324                  .DSABL LSB
                               144B  4325          ;
                               144B  4326 .END
```

```
$$_NSPMSG                       =  00000000
$$_TR3MSG                       =  00000000
$$_TR4MSG                       =  00000000
ACB$B_RMOD                      =  0000000B
ACB$L_KAST                      =  00000018
ACB$L_PID                       =  0000000C
ACK                               000011B3 R      02
ACP$C_STA_F                     =  00000004
ACP$C_STA_H                     =  00000005
ACP$C_STA_I                     =  00000000
ACP$C_STA_N                     =  00000001
ACP$C_STA_R                     =  00000002
ACP$C_STA_S                     =  00000003
ACT$ABORT                         000005B9 RG     02
ACT$CANLNK                        000005B0 RG     02
ACT$RCV_CA                        000003F4 RG     02
ACT$RCV_CC                        000003BB RG     02
ACT$RCV_CI                        0000042A RG     02
ACT$RCV_CR                        00000423 RG     02
ACT$RCV_DATA                      000009F2 RG     02
ACT$RCV_DTACK                     00000624 RG     02
ACT$RCV_DX                        000005CF RG     02
ACT$RCV_LI                        00000780 RG     02
ACT$RCV_LIACK                     0000061D RG     02
ACT$RCV_RTS                       000005B5 RG     02
ACT$RTS_NLT                       00000325 RG     02
ALT_ENTRY                         00000158 R      02
ALT_RCV                           0000015D R      02
ALT_XMT                           00000192 R      02
BACK_PRESSURE                     000009D3 R      02
BLD_ACK_DAT                       0000119E R      02
BLD_ACK_LI                        0000116E R      02
BLD_CA                            0000103C R      02
BLD_CC                            00001020 R      02
BLD_CD                            00000FA5 R      02
BLD_CI                            00000FCE R      02
BLD_DAT                           000010F0 R      02
BLD_DC                            00001045 R      02
BLD_DI                            0000104F R      02
BLD_DISPATCH                      00000F44 R      02
BLD_DTACK                         000010D7 R      02
BLD_DX_COMMON                     0000106F R      02
BLD_LI                            0000108F R      02
BLD_LIACK                         00001087 R      02
BREAK                             00000F8F R      02
BUG$_NETNOSTATE                   ********   X    02
CALC_HXS_LUX                      000008D7 R      02
CALC_HXS_XMT                      000008DA R      02
CANCEL_TIMER                      000013F1 R      02
CHK_INT_AVL                       0000086F R      02
CHK_INT_AVL_R8                    0000086C R      02
CHK_XMT_DONE                      00000752 R      02
CLONE_RCV_CXB                     00000E29 R      02
CLONE_RCV_CXB_1                   00000E26 R      02
CMPL_DISCON                       000005D6 R      02
CNF$_ADVANCE                    =  00000000
CNF$_QUIT                       =  00000002
CNF$_TAKE_CURR                  =  00000003
CNF$_TAKE_PREV                  =  00000001
COM$DRVDELMEM                     ********   X    02
COM$POST                          ********   X    02
COPY_DATA                         00000DB6 R      02
COPY_INT_DATA                     00000217 R      02
CXB$B_CODE                      =  0000000B
CXB$B_R_AREA                      00000039
CXB$B_R_FLG                       00000038
CXB$B_R_NSPTYP                    00000039
CXB$B_TYPE                      =  0000000A
CXB$B_X_NSPTYP                    0000004E
CXB$C_DLL                       =  00000020
CXB$C_HEADER                    =  00000048
CXB$C_OVERHEAD                  =  0000004C
CXB$C_R_LENGTH                  =  0000003C
CXB$C_TRAILER                   =  00000004
CXB$L_LINK                      =  00000010
CXB$L_R_MSG                       0000002C
CXB$L_R_RCB                       00000028
CXB$M_CD_ACK                    =  00000002
CXB$M_CD_XMT                    =  00000001
CXB$T_DLL                       =  00000028
CXB$T_X_DATA                      00000057
CXB$T_X_XPORT                     00000048
CXB$V_CD_XMT                    =  00000000
CXB$W_LENGTH                    =  0000000C
CXB$W_OFFSET                    =  0000000E
CXB$W_R_ADJ                       0000003A
CXB$W_R_BCNT                      00000030
CXB$W_R_DSTNOD                    00000034
CXB$W_R_NSPSEQ                    0000003A
CXB$W_R_PATH                      00000032
CXB$W_R_SRCNOD                    00000036
CXB$W_SIZE                      =  00000008
CXB$W_X_NSPACK                    00000053
CXB$W_X_NSPLOC                    00000051
CXB$W_X_NSPREM                    0000004F
CXB$W_X_NSPSEQ                    00000055
CXB_TO_IRP                        00000A8F R      02
DENIED                            00000F61 R      02
DISCARD                           00000324 R      02
DPT$M_NOUNLOAD                  =  00000004
DYN$C_CXB                       =  0000001B
EX                                00001159 R      02
EXE$ABORTIO                       ********   X    02
EXE$ALONONPAGED                   ********   X    02
EXE$FINISHIOC                     ********   X    02
EXE$FORK                          ********   X    02
EXE$QIORETURN                     ********   X    02
EX_T                              00001152 R      02
FILL_XMT_CXBS                     00000CB8 R      02
FMT_ERROR                         00000324 R      02
GETCTL                            000004B3 R      02
GET_XMT_BUF                       00001214 R      02
GET_XMT_CXB                       000011C1 R      02
ICB$B_ACCESS                    =  0000003C
```

```
ICB$B_LPRNAM                    = 00000014
ICB$B_RPRNAM                    = 00000028
ICB$W_TIM_OCON                  = 00000004
IO$V_INTERRUPT                  = 00000006
IO$V_MULTIPLE                   = 00000008
IPL$_ASTDEL                     = 00000002
IPL$_QUEUEAST                   = 00000006
IRP$B_CXBCNT                    = 00000005  G
IRP$B_QUO                       = 00000004  G
IRP$B_RMOD                      = 0000000B
IRP$C_LENGTH                    = 000000C4
IRP$L_BCNT                      = 00000032
IRP$L_IOQFL                     = 00000000
IRP$L_IOST1                     = 00000038
IRP$L_IOST2                     = 0000003C
IRP$L_MEDIA                     = 00000038
IRP$L_PID                       = 0000000C
IRP$L_SEGVBN                    = 00000048
IRP$L_SES_BUF                   = 00000048  G
IRP$L_SVAPTE                    = 0000002C
IRP$L_UCB                       = 0000001C
IRP$L_WIND                      = 00000018
IRP$M_CHAINED                   = 00000020
IRP$M_COMPLX                    = 00000008
IRP$M_FUNC                      = 00000002
IRP$Q_STATION                   = 00000040
IRP$V_CHAINED                   = 00000005
IRP$V_FUNC                      = 00000001
IRP$W_BCNT                      = 00000032
IRP$W_BOFF                      = 00000030
IRP$W_FUNC                      = 00000020
IRP$W_STS                       = 0000002A
LSB                             = 00000000
LSB$B_R_CXBCNT                  = 00000028
LSB$B_R_CXBQUO                  = 00000029
LSB$B_SPARE                     = 0000002A
LSB$B_STS                       = 0000002B
LSB$B_X_ADJ                     = 0000000B
LSB$B_X_CXBACT                  = 0000000D
LSB$B_X_CXBCNT                  = 0000000F
LSB$B_X_CXBQUO                  = 0000000E
LSB$B_X_PKTWND                  = 0000000C
LSB$B_X_REQ                     = 0000000A
LSB$L_CROSS                     = 0000002C
LSB$L_R_CXB                     = 00000020
LSB$L_R_IRP                     = 0000001C
LSB$L_X_CXB                     = 00000018
LSB$L_X_IRP                     = 00000014
LSB$L_X_PND                     = 00000010
LSB$M_BOM                       = 00000020
LSB$M_EOM                       = 00000040
LSB$M_LI                        = 00000001
LSB$S_LSB                       = 00000030
LSB$S_SPARE                     = 00000004
LSB$S_STS                       = 00000001
LSB$V_BOM                       = 00000005
LSB$V_EOM                       = 00000006
LSB$V_LI                        = 00000000
LSB$V_SPARE                     = 00000001
LSB$W_HAA                       = 00000008
LSB$W_HAR                       = 00000006
LSB$W_HAX                       = 00000026
LSB$W_HNR                       = 00000024
LSB$W_HXS                       = 00000004
LSB$W_LNX                       = 00000002
LSB$W_LUX                       = 00000000
LTB$L_XWB                       = 0000000C
MOVCSFX                           0000058C R      02
MOVCSFX_17                        00000589 R      02
MOVCS_39                          0000053D R      02
MOVPRNAM                          00000553 R      02
MSG$_CONFIRM                    = 00000031
MSG$_INTMSG                     = 00000035
NAK                               000011AE R      02
NDC                             = 00000084
NDC$L_BRC                       = 0000000C
NDC$L_BSN                       = 00000010
NDC$L_PRC                       = 00000014
NDC$L_PSN                       = 00000018
NDC$W_CRC                       = 00000008
NDC$W_CSN                       = 0000000A
NDC$W_RTO                       = 00000006
NET$ACK_XMT_SEGS                  000006E7 RG     02
NET$ALONONPAGED                   ******** GX     02
NET$ALTENTRY                      000000EB RG     02
NET$AT_RCVMSG                     00000004 R      02
NET$CCS_IOSTAT                    0000123A RG     02
NET$CHK_X_IDLE                    ********  X     02
NET$CMPL_ACC                      ********  X     02
NET$COMPLEX_EV                    ********  X     02
NET$CREATE_XWB                    ********  X     02
NET$C_ACT_TIMER                 = 0000001E
NET$C_DR_ABORT                  = 00000009
NET$C_DR_ACCESS                 = 00000022
NET$C_DR_CONF                   = 0000002A
NET$C_DR_EXIT                   = 00000026
NET$C_DR_FMT                    = 00000005
NET$C_DR_INVALID                  ********  X     02
NET$C_DR_NOLINK                 = 00000029
NET$C_DR_NOPATH                 = 00000027
NET$C_DR_PROTCL                 = 00000007
NET$C_DR_RSU                    = 00000001
NET$C_DR_SEGSIZ                 = 00000025
NET$C_DR_ZERO                   = 00000017
NET$C_EFN_ASYN                  = 00000002
NET$C_EFN_WAIT                  = 00000001
NET$C_IPL                       = 00000008
NET$C_MAXACCFLD                 = 00000027
NET$C_MAXLINNAM                 = 0000000F
NET$C_MAXLNK                    = 000003FF
NET$C_MAXNODNAM                 = 00000006
NET$C_MAXOBJNAM                 = 0000000C
NET$C_MAX_AREAS                 = 0000003F
NET$C_MAX_LINES                 = 00000040
```

```
NET$C_MAX_NCB                   = 0000006E
NET$C_MAX_NODES                 = 000003FF
NET$C_MAX_OBJ                   = 000000FF
NET$C_MAX_WQE                   = 00000014
NET$C_MINBUFSIZ                 = 000000C0
NET$C_TID_ACT                   = 00000003
NET$C_TID_RUS                   = 00000001
NET$C_TID_XRT                   = 00000002
NET$C_TRCTL_CEL                 = 00000002
NET$C_TRCTL_OVR                 = 00000005
NET$C_UTLBUFSIZ                 = 00001000
NET$DEALLOCATE                    ********    X   02
NET$DRAIN_R_IRPCXB                00000E6B RG     02
NET$DRAIN_R_LSBCXB                00000E7F R      02
NET$EVENT                         ********    X   02
NET$FDT_RCV                       00000111 RG     02
NET$FDT_XMT                       0000011E RG     02
NET$FORK                          ********    X   02
NET$GL_OFF_DPTFLG                 ********    X   02
NET$GL_WORKBITS                   ********    X   02
NET$IO_STATUS                     00001263 RG     02
NET$KAST                          00000C56 RG     02
NET$MAP_R_REASON                  ********    X   02
NET$MARK_LINK                     000013B4 RG     02
NET$MOV_CSTR                      ********    X   02
NET$MOV_USTR                      ********    X   02
NET$M_MAXLNKMSK                 = 000003FF
NET$PIG_ACK                       00000607 R      02
NET$PRE_EMPT                      ********    X   02
NET$PURG_RUN                      ********    X   02
NET$QAST                          00000C12 RG     02
NET$QUE_XWB                       ********    X   02
NET$RCV_DONE                      00000B82 RG     02
NET$RESET_TIMER                   00001433 RG     02
NET$RET_SLOT                      ********    X   02
NET$SCH_MSG                       ********    X   02
NET$SEND_CS_MBX                   ********    X   02
NET$SETUP_RUN                     00000028 RG     02
NET$TIMER                         0000126D RG     02
NET$UNSOL_INTR                    00000242 RG     02
NET$XMT_DONE                      00000766 RG     02
NET$XWB_LOCLNK                    ********    X   02
NETEVT$_CA                        ********    X   02
NETEVT$_CC                        ********    X   02
NETEVT$_CI                        ********    X   02
NETEVT$_DATA                      ********    X   02
NETEVT$_DC                        ********    X   02
NETEVT$_DI                        ********    X   02
NETEVT$_DSCLNK                    ********    X   02
NETEVT$_DTACK                     ********    X   02
NETEVT$_INT                       ********    X   02
NETEVT$_LIACK                     ********    X   02
NETEVT$_LS                        ********    X   02
NETEVT$_MBXERR                    ********    X   02
NETEVT$_PH2CCS                    ********    X   02
NETEVT$_PROERR                    ********    X   02
NETEVT$_RTS                       ********    X   02
NEW_DATA_FLOW                     000008A9 R      02
NEW_RCV_IRP                       0000096C R      02
NONE                              00000FA2 R      02
NOT_NEXT                          000009BA R      02
NO_BUF                            000009C1 R      02
NO_RSRC                           00000325 R      02
NSP$$$_QUAL_ACK                 = 00000000
NSP$$$_QUAL_ALTFLW              = 00000000
NSP$$$_QUAL_DATA                = 00000000
NSP$$$_QUAL_FLW                 = 00000000
NSP$$$_QUAL_INF                 = 00000000
NSP$$$_QUAL_MSG                 = 00000000
NSP$$$_QUAL_SRV                 = 00000000
NSP$B_ADJ_XPW                     00000000 RG     02
NSP$B_MAX_RBF                     00000002 RG     02
NSP$B_MAX_XPW                     00000001 RG     02
NSP$B_R_CXBTHR                    00000003 RG     02
NSP$C_ADJ_XPW                   = 00000020
NSP$C_EXT_LNK                   = 0000001E
NSP$C_FLW_DATA                  = 00000000
NSP$C_FLW_INT                   = 00000001
NSP$C_FLW_NOP                   = 00000000
NSP$C_FLW_XOFF                  = 00000001
NSP$C_FLW_XON                   = 00000002
NSP$C_HSZ_ACK                   = 00000007
NSP$C_HSZ_CA                    = 00000003
NSP$C_HSZ_CC                    = 00000064
NSP$C_HSZ_CD                    = 000000F0
NSP$C_HSZ_CI                    = 000000F0
NSP$C_HSZ_DATA                  = 00000009
NSP$C_HSZ_DC                    = 00000016
NSP$C_HSZ_DI                    = 00000016
NSP$C_HSZ_INT                   = 00000009
NSP$C_HSZ_LS                    = 00000009
NSP$C_INF_V31                   = 00000001
NSP$C_INF_V32                   = 00000000
NSP$C_INF_V33                   = 00000002
NSP$C_INF_V40                   = 00000002
NSP$C_MAXHDR                    = 00000009
NSP$C_MAX_DELAY                 = 00000014
NSP$C_MAX_RBF                   = 00000007
NSP$C_MAX_R_CXB                 = 00000007
NSP$C_MAX_XPW                   = 00000028
NSP$C_MSG_CA                    = 00000024
NSP$C_MSG_CC                    = 00000028
NSP$C_MSG_CI                    = 00000018
NSP$C_MSG_CR                    = 00000068
NSP$C_MSG_DATA                  = 00000000
NSP$C_MSG_DC                    = 00000048
NSP$C_MSG_DI                    = 00000038
NSP$C_MSG_DTACK                 = 00000004
NSP$C_MSG_INT                   = 00000030
NSP$C_MSG_LIACK                 = 00000014
NSP$C_MSG_LS                    = 00000010
NSP$C_R_CXBTHR                  = 00000005
NSP$C_SRV_MFC                   = 00000002
NSP$C_SRV_NFC                   = 00000000
```

```
NSP$C_SRV_REQ                   =  00000001
NSP$C_SRV_SFC                   =  00000001
NSP$M_ACK_NAK                   =  00001000
NSP$M_ACK_NUM                   =  00000FFF
NSP$M_ACK_VALID                 =  00008000
NSP$M_DATA_BOM                  =  00000020
NSP$M_DATA_EOM                  =  00000040
NSP$M_DATA_NAR                  =  00000080
NSP$M_DATA_OVFW                 =  00000080
NSP$M_FLW_CHAN                  =  0000000C
NSP$M_FLW_DRV                   =  000000F0
NSP$M_FLW_INT                   =  00000020
NSP$M_FLW_INUSE                 =  00000010
NSP$M_FLW_LISUB                 =  00000004
NSP$M_FLW_MODE                  =  00000003
NSP$M_FLW_SP1                   =  00000008
NSP$M_FLW_SP2                   =  00000040
NSP$M_FLW_SP3                   =  00000080
NSP$M_FLW_XOFF                  =  00000001
NSP$M_FLW_XON                   =  00000002
NSP$M_INF_VER                   =  00000003
NSP$M_MSG_INT                   =  00000020
NSP$M_MSG_LI                    =  00000010
NSP$M_SEQ_NAR                   =  00004000
NSP$M_SRV_01                    =  00000003
NSP$M_SRV_EXT                   =  00000080
NSP$M_SRV_FLW                   =  0000000C
NSP$M_SRV_REQ                   =  000000F3
NSP$M_SRV_SP1                   =  00000070
NSP$R_QUAL                      =  00000000
NSP$SOLICIT                        00000E93 RG     02
NSP$S_ACK_NUM                   =  0000000C
NSP$S_ACK_SP2                   =  00000002
NSP$S_DATA_SP                   =  00000005
NSP$S_FLW_CHAN                  =  00000002
NSP$S_FLW_DRV                   =  00000004
NSP$S_FLW_MODE                  =  00000002
NSP$S_INF_VER                   =  00000002
NSP$S_MSG_SP1                   =  00000004
NSP$S_NSPMSG                    =  00000005
NSP$S_QUAL                      =  00000005
NSP$S_QUAL_ACK                  =  00000002
NSP$S_QUAL_ALTFLW               =  00000001
NSP$S_QUAL_DATA                 =  00000001
NSP$S_QUAL_FLW                  =  00000001
NSP$S_QUAL_INF                  =  00000001
NSP$S_QUAL_MSG                  =  00000005
NSP$S_QUAL_SRV                  =  00000001
NSP$S_SRV_01                    =  00000002
NSP$S_SRV_FLW                   =  00000002
NSP$S_SRV_SP1                   =  00000003
NSP$V_ACK_NAK                   =  0000000C
NSP$V_ACK_NUM                   =  00000000
NSP$V_ACK_SP2                   =  0000000D
NSP$V_ACK_VALID                 =  0000000F
NSP$V_ACK_XCH                   =  0000000D
NSP$V_DATA_BOM                  =  00000005
NSP$V_DATA_EOM                  =  00000006
NSP$V_DATA_NAR                  =  00000007
NSP$V_DATA_OVFW                 =  00000007
NSP$V_DATA_SP                   =  00000000
NSP$V_FLW_CHAN                  =  00000002
NSP$V_FLW_DRV                   =  00000004
NSP$V_FLW_INT                   =  00000005
NSP$V_FLW_INUSE                 =  00000004
NSP$V_FLW_LISUB                 =  00000002
NSP$V_FLW_MODE                  =  00000000
NSP$V_FLW_SP1                   =  00000003
NSP$V_FLW_SP2                   =  00000006
NSP$V_FLW_SP3                   =  00000007
NSP$V_FLW_XOFF                  =  00000000
NSP$V_FLW_XON                   =  00000001
NSP$V_INF_VER                   =  00000000
NSP$V_MSG_INT                   =  00000005
NSP$V_MSG_LI                    =  00000004
NSP$V_MSG_SP1                   =  00000000
NSP$V_SEQ_NAR                   =  0000000E
NSP$V_SRV_01                    =  00000000
NSP$V_SRV_EXT                   =  00000007
NSP$V_SRV_FLW                   =  00000002
NSP$V_SRV_SP1                   =  00000004
NSP$W_DSTLNK                    =  00000001
NSP$W_SRCLNK                    =  00000003
OVF                                00009A6 R      02
P1                              =  00000000
P2                              =  00000004
PR$_IPL                         =  00000012
PROC_ACK                           0000063F R      02
PROC_DTACK                         0000068B R      02
PROC_LIACK                         000006B3 R      02
PRS_CHR                            00000453 R      02
QRL$SETUP_CHAN                     ******** *    X 02
QUICK_SOL                          00000EAE R      02
R7_CXB_TO_IRP                      00000A8C R      02
RCB$B_ECL_RFLW                  =  00000062
RCB$L_ACP_UCB                   =  00000014
RCB$L_PTR_LTB                   =  00000024
RCB$W_MCOUNT                    =  00000054
RCVMAP_B_EVT                    =  00000002
RCVMAP_B_MSG                    =  00000000
RCVMAP_B_SIZ                    =  00000001
RCVMAP_C_END                    =  0000FFFF
RCV_COMMON                         00000172 R      02
RCV_COPY                           00000BED R      02
RCV_COPY1                          00000BD1 R      02
RCV_COPY2                          00000BCE R      02
RCV_DONE                           00000B89 R      02
RCV_IRP                            00000993 R      02
RCV_MSG                            0000024F R      02
REASON_W_DR                        ******** *    X 02
REASON_W_MBX                       ******** *    X 02
REASON_W_SS                        ******** *    X 02
RESET                              00001377 R      02
RESET_TIMER                        00001433 R      02
```

```
RTS_NLT                         00000325 R      02
RW_FDT                          00000122 R      02
SCA$QAST                        ******** X      02
SETUP_LSB                       000000E0 R      02
SET_TIMER_RUN                   00001427 R      02
SET_X                           000005BB R      02
SHRINK_XPW                      00000886 R      02
SIZ...                        = 00000001
SS$_ABORT                     = 0000002C
SS$_ACCVIO                    = 0000000C
SS$_BUFFEROVF                 = 00000601
SS$_DATAOVERUN                = 00000838
SS$_FILNOTACC                 = 000000AC
SS$_LINKABORT                 = 000020E4
SS$_NOMBX                     = 00000274
SS$_NORMAL                    = 00000001
SS$_TOOMUCHDATA               = 0000029C
TIMED_SEG_ACKED                 000013C0 R      02
TIMEOUT                         000012ED R      02
TQE$B_RQTYPE                  = 0000000B
TQE$M_REPEAT                  = 00000004
TR$C_MAXHDR                   = 0000001C
TR$C_NI_ALLEND1               = 040000AB
TR$C_NI_ALLEND2               = 00000000
TR$C_NI_ALLROU1               = 030000AB
TR$C_NI_ALLROU2               = 00000000
TR$C_NI_PREFIX                = 000400AA
TR$C_NI_PROT                  = 00000360
TR$C_PRI_ECL                  = 0000001F
TR$C_PRI_RTHRU                = 0000001F
TR$KILL_LOC_LPD                 ******** X      02
TR$SOLICIT                      ******** X      02
TR$TEST_REACH                   ******** X      02
TR$TIMER                        ******** X      02
TR3$$$_QUAL_MSG               = 00000000
TR3$$$_QUAL_RTFLG             = 00000000
TR3$C_ASZ_DATA                = 00000006
TR3$C_MSG_DATA                = 00000002
TR3$C_MSG_HELLO               = 00000005
TR3$C_MSG_INIT                = 00000001
TR3$C_MSG_NOP2                = 00000008
TR3$C_MSG_ROUT                = 00000007
TR3$C_MSG_STR2                = 00000058
TR3$C_MSG_VERF                = 00000003
TR3$M_MSG_CTL                 = 00000001
TR3$M_MSG_RTH                 = 00000002
TR3$M_RTFLG_PH2               = 00000040
TR3$M_RTFLG_RQR               = 00000008
TR3$M_RTFLG_RTS               = 00000010
TR3$R_QUAL                    = 00000000
TR3$S_QUAL                    = 00000001
TR3$S_QUAL_MSG                = 00000001
TR3$S_QUAL_RTFLG              = 00000001
TR3$S_RTFLG_012               = 00000003
TR3$S_TR3MSG                  = 00000001
TR3$V_MSG_CTL                 = 00000000
TR3$V_MSG_RTH                 = 00000001
TR3$V_RTFLG_012               = 00000000
TR3$V_RTFLG_5                 = 00000005
TR3$V_RTFLG_7                 = 00000007
TR3$V_RTFLG_PH2               = 00000006
TR3$V_RTFLG_RQR               = 00000003
TR3$V_RTFLG_RTS               = 00000004
TR4$$$_QUAL_ADDR              = 00000000
TR4$$$_QUAL_RTFLG             = 00000000
TR4$$$_QUAL_SCLASS            = 00000000
TR4$C_BCE_MID1                = 040000AB
TR4$C_BCE_MID2                = 00000000
TR4$C_BCR_MID1                = 030000AB
TR4$C_BCR_MID2                = 00000000
TR4$C_BCT3MULT                = 00000008
TR4$C_END_NODE                = 00000003
TR4$C_HIORD                   = 000400AA
TR4$C_HSZ_DATA                = 00000015
TR4$C_MSG_BCEHEL              = 0000000D
TR4$C_MSG_BCRHEL              = 0000000B
TR4$C_MSG_LDATA               = 00000006
TR4$C_MSG_RDATA               = 00000002
TR4$C_PRO_TYPE                = 00000360
TR4$C_RTR_LVL1                = 00000002
TR4$C_RTR_LVL2                = 00000001
TR4$C_T3MULT                  = 00000002
TR4$C_VER_HIB                 = 00000000
TR4$C_VER_LOWW                = 00000002
TR4$M_ADDR_AREA               = 0000FC00
TR4$M_ADDR_DEST               = 000003FF
TR4$M_RTFLG_INI               = 00000020
TR4$M_RTFLG_LNG               = 00000004
TR4$M_RTFLG_RQR               = 00000008
TR4$M_RTFLG_RTS               = 00000010
TR4$R_QUAL                    = 00000000
TR4$S_ADDR_AREA               = 00000006
TR4$S_ADDR_DEST               = 0000000A
TR4$S_QUAL                    = 00000002
TR4$S_QUAL_ADDR               = 00000002
TR4$S_QUAL_RTFLG              = 00000001
TR4$S_QUAL_SCLASS             = 00000001
TR4$S_RTFLG_01                = 00000002
TR4$S_RTFLG_VER               = 00000002
TR4$S_SCLASS_57               = 00000003
TR4$S_TR4MSG                  = 00000002
TR4$V_ADDR_AREA               = 0000000A
TR4$V_ADDR_DEST               = 00000000
TR4$V_RTFLG_01                = 00000000
TR4$V_RTFLG_INI               = 00000005
TR4$V_RTFLG_LNG               = 00000002
TR4$V_RTFLG_RQR               = 00000003
TR4$V_RTFLG_RTS               = 00000004
TR4$V_RTFLG_VER               = 00000006
TR4$V_SCLASS_1                = 00000001
TR4$V_SCLASS_57               = 00000005
TR4$V_SCLASS_BC               = 00000004
TR4$V_SCLASS_LS               = 00000002
TR4$V_SCLASS_METR             = 00000000
```

```
TR4$V_SCLASS_SUBA               = 00000003
T_O_CC                            00001302 R      02
T_O_CI                            00001302 R      02
T_O_DI                            00001302 R      02
T_O_RUN                           0000130A R      02
UCB$B_FIPL                      = 0000000B
UCB$L_VCB                       = 00000034
UNK_MSG                           00000324 R      02
UPD_PROGRESS                      00001389 R      02
VA$M_BYTE                       = 000001FF
XCHAN                             00000634 R      02
XMT_COMMON                        00000196 R      02
XMT_COPY                          00000D36 R      02
XMT_COPY1                         00000D00 R      02
XMT_INT_CO                        000001CB R      02
XMT_RCV_CO                        000001D2 R      02
XMT_REQ_DONE                      0000072B R      02
XMT_REQ_DONE_OK                   0000072F R      02
XWB                             = 00000000
XWB$$                             ********    X   02
XWB$B_ACCESS                    = 0000000B
XWB$B_ADJ_INX                     ********    X   02
XWB$B_DATA                      = 0000005B
XWB$B_FIPL                      = 0000001F
XWB$B_LOGIN                     = 000000CC
XWB$B_LPRNAM                    = 000000A4
XWB$B_PRO                       = 0000005A
XWB$B_RID                       = 0000006F
XWB$B_RPRNAM                    = 000000B8
XWB$B_SP3                       = 0000006E
XWB$B_STA                       = 0000001E
XWB$B_TYPE                      = 0000000A
XWB$B_X_FLW                     = 0000006C
XWB$B_X_FLWCNT                  = 0000006D
XWB$C_COMLNG                    = 000000A4
XWB$C_CONLNG                    = 00000112
XWB$C_DATA                      = 00000010
XWB$C_LOGIN                     = 00000040
XWB$C_LPRNAM                    = 00000014
XWB$C_NDC_LNG                   = 00000020
XWB$C_NUMSTA                    = 00000008
XWB$C_RID                       = 00000010
XWB$C_RPRNAM                    = 00000014
XWB$C_STA_CAR                   = 00000002
XWB$C_STA_CCS                   = 00000004
XWB$C_STA_CIR                   = 00000003
XWB$C_STA_CIS                   = 00000001
XWB$C_STA_CLO                   = 00000000
XWB$C_STA_DIR                   = 00000006
XWB$C_STA_DIS                   = 00000007
XWB$C_STA_RUN                   = 00000005
XWB$L_DEA_IRP                   = 00000104
XWB$L_FPC                       = 00000020
XWB$L_FR3                       = 00000024
XWB$L_FR4                       = 00000028
XWB$L_ICB                       = 0000010C
XWB$L_IRP_ACC                   = 00000080
XWB$L_LINK                      = 0000002C
XWB$L_ORGUCB                    = 00000010
XWB$L_PID                       = 00000034
XWB$L_PTR_RTHD                    ********    X   02
XWB$L_VCB                       = 00000030
XWB$L_WLBL                      = 00000004
XWB$L_WLFL                      = 00000000
XWB$M_FLG_BREAK                 = 00000001
XWB$M_FLG_CLO                   = 00000200
XWB$M_FLG_IAVL                  = 00001000
XWB$M_FLG_SCD                   = 00000100
XWB$M_FLG_SDACK                 = 00000008
XWB$M_FLG_SDFL                  = 00004000
XWB$M_FLG_SDT                   = 00000080
XWB$M_FLG_SIACK                 = 00000004
XWB$M_FLG_SIFL                  = 00002000
XWB$M_FLG_SLI                   = 00000010
XWB$M_FLG_TBPR                  = 00000800
XWB$M_FLG_WBP                   = 00000040
XWB$M_FLG_WBUF                  = 00000002
XWB$M_FLG_WDAT                  = 00000400
XWB$M_FLG_WHGL                  = 00000020
XWB$M_PRO_CCA                   = 00000008
XWB$M_PRO_NAR                   = 00000010
XWB$M_PRO_NFC                   = 00000001
XWB$M_PRO_PH2                   = 00000004
XWB$M_PRO_SFC                   = 00000002
XWB$M_STS_ASTPND                = 00000400
XWB$M_STS_ASTREQ                = 00000800
XWB$M_STS_CON                   = 00000010
XWB$M_STS_DIS                   = 00000008
XWB$M_STS_DTNAK                 = 00000100
XWB$M_STS_LINAK                 = 00000200
XWB$M_STS_NDC                   = 00001000
XWB$M_STS_OVF                   = 00000080
XWB$M_STS_RBP                   = 00000040
XWB$M_STS_SOL                   = 00000004
XWB$M_STS_TID                   = 00000001
XWB$M_STS_TLI                   = 00000002
XWB$M_STS_TMO                   = 00000020
XWB$Q_FORK                      = 00000014
XWB$Q_FREE_CXB                  = 00000118
XWB$R_CON_BLK                   = 000000A4
XWB$R_RUN_BLK                   = 000000A4
XWB$S_                          = 00000006
XWB$S_COMLNG                    = 0000006E
XWB$S_CON_BLK                   = 0000006E
XWB$S_DATA                      = 00000010
XWB$S_DT                        = 00000030
XWB$S_FLG                       = 00000002
XWB$S_FORK                      = 00000008
XWB$S_FREE_CXB                  = 00000008
XWB$S_LI                        = 00000030
XWB$S_LOGIN                     = 0000003F
XWB$S_LPRNAM                    = 00000013
XWB$S_NDC                       = 00000020
XWB$S_PRO                       = 00000001
```

```
XWB$S_RID                       = 00000010
XWB$S_RPRNAM                    = 00000013
XWB$S_RUN_BLK                   = 00000064
XWB$S_STS                       = 00000002
XWB$S_XWB                       = 00000120
XWB$T_                          = 00000112
XWB$T_DATA                      = 0000005C
XWB$T_DT                        = 000000A4
XWB$T_LI                        = 000000D4
XWB$T_LOGIN                     = 000000CD
XWB$T_LPRNAM                    = 000000A5
XWB$T_RID                       = 00000070
XWB$T_RPRNAM                    = 000000B9
XWB$V_FLG_BREAK                 = 00000000
XWB$V_FLG_CLO                   = 00000009
XWB$V_FLG_IAVL                  = 0000000C
XWB$V_FLG_SCD                   = 00000008
XWB$V_FLG_SDACK                 = 00000003
XWB$V_FLG_SDFL                  = 0000000E
XWB$V_FLG_SDT                   = 00000007
XWB$V_FLG_SIACK                 = 00000002
XWB$V_FLG_SIFL                  = 0000000D
XWB$V_FLG_SLI                   = 00000004
XWB$V_FLG_TBPR                  = 0000000B
XWB$V_FLG_WBP                   = 00000006
XWB$V_FLG_WBUF                  = 00000001
XWB$V_FLG_WDAT                  = 0000000A
XWB$V_FLG_WHGL                  = 00000005
XWB$V_PRO_CCA                   = 00000003
XWB$V_PRO_NAR                   = 00000004
XWB$V_PRO_NFC                   = 00000000
XWB$V_PRO_PH2                   = 00000002
XWB$V_PRO_SFC                   = 00000001
XWB$V_STS_ASTPND                = 0000000A
XWB$V_STS_ASTREQ                = 0000000B
XWB$V_STS_CON                   = 00000004
XWB$V_STS_DIS                   = 00000003
XWB$V_STS_DTNAK                 = 00000008
XWB$V_STS_LINAK                 = 00000009
XWB$V_STS_NDC                   = 0000000C
XWB$V_STS_OVF                   = 00000007
XWB$V_STS_RBP                   = 00000006
XWB$V_STS_SOL                   = 00000002
XWB$V_STS_TID                   = 00000000
XWB$V_STS_TLI                   = 00000001
XWB$V_STS_TMO                   = 00000005
XWB$W_CI_PATH                   = 00000110
XWB$W_DELAY                     = 0000004E
XWB$W_DLY_FACT                  = 00000056
XWB$W_DLY_WGHT                  = 00000058
XWB$W_ELAPSE                    = 0000004A
XWB$W_FLG                       = 0000001C
XWB$W_LOCLNK                    = 0000003E
XWB$W_LOCSIZ                    = 00000040
XWB$W_PATH                      = 00000038
XWB$W_PROGRESS                  = 00000052
XWB$W_REFCNT                    = 0000000C
XWB$W_REMLNK                    = 0000003C
XWB$W_REMNOD                    = 0000003A
XWB$W_REMSIZ                    = 00000042
XWB$W_RETRAN                    = 00000054
XWB$W_R_REASON                  = 00000044
XWB$W_SIZE                      = 00000008
XWB$W_STS                       = 0000000E
XWB$W_TIMER                     = 00000050
XWB$W_TIM_ID                    = 00000048
XWB$W_TIM_INACT                 = 0000004C
XWB$W_X_REASON                  = 00000046
XWB$Z_NDC                       = 00000084
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000057 (87.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
$$$115_DRIVER	0000144B (5195.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	24	00:00:00.11	00:00:01.03
Command processing	149	00:00:01.13	00:00:05.60
Pass 1	1261	00:00:38.34	00:01:14.62
Symbol table sort	4	00:00:05.12	00:00:09.79
Pass 2	1301	00:00:11.55	00:00:26.39
Symbol table output	1	00:00:00.57	00:00:01.11
Psect synopsis output	3	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	2745	00:00:56.87	00:01:58.57

The working set limit was 1350 pages.
214753 bytes (420 pages) of virtual memory were used to buffer the intermediate code.
There were 170 pages of symbol table space allocated to hold 3045 non-local and 282 local symbols.
4326 source lines were read in Pass 1, producing 31 object records in Pass 2.
76 pages of virtual memory were used to define 58 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SHRLIB]NMALIBRY.MLB;1	0
_255DUA28:[SHRLIB]EVCDEF.MLB;1	0
_255DUA28:[NETACP.OBJ]NETDRV.MLB;1	3
_255DUA28:[NETACP.OBJ]NET.MLB;1	10
_255DUA28:[SYS.OBJ]LIB.MLB;1	24
_255DUA28:[SYSLIB]STARLET.MLB;2	10
TOTALS (all libraries)	47

3177 GETS were required to define 47 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:NETDRVNSP/OBJ=OBJ$:NETDRVNSP MSRC$:NETDRVNSP/UPDATE=(ENH$:NETDRVNSP)+EXECML$/LIB+LIB$:NET/LIB+LIB$:NETDRV/LIB+SHRLIB$

NETDRVQRL
LIS

NETDRVSES
LIS

NETDRVNSP
LIS